VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2013 Radoslav Štrba

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace softwarové podpory SCRUM
SCRUM Software Support Application

2013 **Bc. Radoslav Štrba**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Radoslav Štrba**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Aplikace softwarové podpory SCRUM
SCRUM Software Support Application

Zásady pro vypracování:

Cílem práce je nastudovat pokročilé softwarové procesy. Student se následně zaměří na agilní metodiky nejčastěji používané v komerčních oblastech vývoje softwaru. Následně pak implementuje nástroj pro podporu SCRUMu v komerční oblasti. Vývoj tohoto nástroje bude zaměřen primárně na optimalizaci softwarových procesů v komerčních společnostech a bude pokrývat vše od evidence týmů a jejich členů, přes přidělování rolí, plánování a přiřazování úkolů a dalších prvků, které SCRUM ve své podstatě nabízí.

Specifické cíle práce budou:
1. Nastudovat agilní metodiky vývoje softwaru a srovnat je s dalšími existujícími.
2. Zaměřit se na SCRUM a jeho detailnější popis - vlastnosti, výhody, používané nástroje (SCRUM board, burndown, logy apod.).
3. Zvolit prvky SCRUMu a doplňky pro podporu softwarového procesu pro firmu AstrumQ Interactive, s.r.o. a jí podobné. Student navrhne integrace a vylepšení pro tyto nástroje.
4. Implementovat aplikaci pro podporu SCRUMu, která umožní vše, co vyžaduje proces SCRUM od editace týmů a jejich členů, přes úkoly a jejich plánování a realizaci (kompatibilní s mobilními zařízeními jako iPad, Galaxy apod.).

Seznam doporučené odborné literatury:

[1] MACDONALD, M., FREEMAN, A., SZPUSZTA, M. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně, kniha 1. Brno: Zoner Press, 2011. ISBN 978-80-7413-131-8
[2] MACDONALD, M., FREEMAN, A., SZPUSZTA, M. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně, kniha 2. Brno: Zoner Press, 2011. ISBN 978-80-7413-145-5
[3] RESIG, John. jQuery: Kuchařka programátora. Brno: Computer Press, 2010. ISBN 978-80-251-3152-7
[4] KOŠINÁR, M.: Design and Utilization of Knowledge Bases for Software Processes, 2010. Diplomová práce na FEI VŠB TU Ostrava

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michal Košinár**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlásenie študenta

Prehlasujem že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne parametre a publikácie, z ktorých som čerpal.

Dňa: *17.4.2013* ……..……

podpis študenta

Poďakovanie

Rád by som poďakoval vedúcemu mojej diplomovej práce Michalovi Košinárovi, za jeho odborný prístup, cenné rady a motiváciu. Poďakovanie taktiež patrí Jakubovi Štolfovi a Svätoplukovi Štolfovi, za ich kritiku, konzultácie a možnosť spolupráce so spoločnosťou SCOVECO, s.r.o. Ďakujem aj Jánu Bičiště, za konzultácie a možnosť spolupráce so spoločnosťou STAPRO, s.r.o.

Abstrakt

Zmena je aspekt na ktorom sú založené moderné metódy vývoja softwaru. V súčasnej dobe je potrebné aplikovať zmeny požiadaviek aj v neskorších fázach vývoja. Tradičné metódy vývoja, ako napríklad UP alebo RUP v komerčnej sfére veľmi nefungujú. Ich hlavnou myšlienkou je naplánovať všetko na začiatku a nepočíta sa s prípadnými zmenami v neskorších fázach vývoja. Cieľom tejto práce je naštudovanie pokročilých softwarových procesov. Zameriame sa na agilné metódy, najčastejšie používané v komerčnej sfére vývoja softwaru. Požiadavky na software sa stávajú čoraz viac komplikovanejšie. Dnešné softwarové systémy sú komplexnejšie a rozsiahlejšie. Toto sú dôvody, prečo nie je možné ďalej efektívne vyvíjať, bez podpory vhodných nástrojov. Na základe nadobudnutých znalostí bol vyvinutý softwarový nástroj pre podporu SCRUM. Nástroj bol implementovaný v softwarových spoločnostiach na zlepšenie riadenia vývoja, založeného na agilných metódach a na optimalizáciu navrhnutého softwarového procesu. Konkrétne, pre plánovanie času a rozpočtu projektov, automaticky zobrazované hodinové výpisy práce zamestnancov a prehľad aktuálnych stavov konkrétnych úloh.

Kľúčové slova

aplikácia softvérovej podpory, SCRUM board, agilné metodológie, biznis procesy, softvérové procesy

Abstract

Change is an aspect that is anticipant in agile-based development. There is a demand for new models that allow developers to change the plans even later on. In the present situation, traditional methods, like UP or RUP does not work. However, at the very core of traditional methods (UP, RUP), is the idea of planning everything at the beginning of the project. Goal of this thesis is study advanced software processes. We focus on agile methodologies mostly used in commercial software development. Software requirements are becoming more and more complicated. Software systems of today are characterized by increasing complexity and size. Therefore can no longer be developed feasibly without the aid of supporting appropriate tools. Based on gathered knowledge was developed a software tool for SCRUM support. The tool was subsequently implemented in software companies to improve project management with agile software development and optimize designed software process. Specifically plan time and cost of projects, automatically report hourly time sheets of employees and an overview of the current status of specific tasks.

Key words

SCRUM Software Support Application, agile methodologies, business processes, software processes

Zoznam použitých skratiek

Skratka	Význam
AJAX	Asynchronous JavaScript and XML
ASP	Active Server Pages
BUC	Business Use Cases
DES	Discrete-Event Simulation
EF	Entity Framework
EPC	Event-driven Process Chain
JSON	JavaScript Object Notation
LINQ	Language Integrated Query
MVC	Model – View – Controller
ORM	Object Rational Mapping
RUP	Rational Unified Process
SQL	Structured Query Language
TDD	Test-Driven Development
UML	Unified Modeling Language
UP	Unified Process
XML	Extensible Markup Language
XP	Extreme Programming

Obsah

1 Úvod

Agilné metódy sa stali v komerčnej sfére vývoja softwaru veľmi obľúbenými. Jedným z hlavných dôvodov ich používania, je schopnosť prijať zmeny a zareagovať na ne oveľa efektívnejšie, ako by to šlo pri používaní tradičných rigoróznych metód (UP, RUP). Práve zmena je dôležitým aspektom, na ktorom sú založené agilné metodológie vývoja softwaru. Dôležitú úlohu pri používaní agilných metód zohráva komunikácia. Komunikácia, sledovanie pokroku a zmien sú dôležité faktory pre úspešne riadenie softwarových projektov. Práve toto sú kľúčové dôvody pre používanie podporných softwarových nástrojov pri riadení vývoja projektov.

1.1 Rozsah a ciele diplomovej práce

Stručný popis špecifických cieľov diplomovej práce, ktoré boli postupne splnené.

1. Naštudovanie agilných prístupov a ich porovnanie s ďalšími existujúcimi.
2. Návrh softwarových Scrum procesov a podporných procesov pre prácu s aplikáciou.
3. Analýza a návrh aplikácie pre podporu softwarových procesov.
4. Implementácia aplikácie.
5. Zaškolenie kľúčových zástupcov spoločností pre oblasť metodiky vývoja SW SCRUM, spolu s metodikami o správnom dodržiavaní softwarového procesu, ktorý bol navrhnutý.

Ako doplnková časť práce bola vyvinutá knižnica pre simuláciu softwarového procesu, spolu s natívnym grafickým rozhraním. Podrobnejšie je vývoj tejto knižnice a popis metódy zvolenej pre simuláciu softwarového procesu rozobratá v šiestej kapitole.

1.2 Popis hlavných kapitol práce

Táto práca sa zaoberá implementáciou nástroja, určeného pre podporu a optimalizáciu softwarových procesov v komerčných spoločnostiach a taktiež návrhom softwarových procesov, pre konkrétne softwarové spoločnosti.

Druhá kapitola obsahuje teoretickú časť práce, ktorá je venovaná teoretickému popisu agilných metodológií vývoja softvéru a ich porovnanie s tradičnými rigoróznymi metódami. Pozornosť je venovaná hlavne frameworku SCRUM, popisu jeho vlastností a výhod. Následne sú popísané funkcie a vlastnosti on-line nástrojov pre podporu procesov, založených na agilných metodológiách vývoja softvéru.

Tretia, skrátená kapitola sa zaoberá motiváciou používania softwarových nástrojov pre podporu procesov. Motivácia je zameraná na výhody používania nástrojov, aj napriek tomu, že Scrum prístup takúto podporu vo svojej podstate nevyžaduje.

Vo štvrtej kapitole je predstavený návrh riešenia, ktorý spočíva špecifikácii zadania, špecifikácii požiadaviek a následne návrhu softwarový procesov. Ďalej je predstavený návrh podporných procesov pre prácu s aplikáciou a taktiež Scrum procesy pre spoločnosti

STAPRO, SCOVECO a satelitnú platformu Skylink zo skupiny M7. Po návrhu softwarových procesov je v tejto kapitole popísaná analýza a návrh aplikácie pre podporu týchto procesov.

V piatej kapitole sú podrobnejšie popísané zaujímavé riešenia a technológie použité pri implementácií aplikácie. Taktiež sú tu predstavené základné vlastnosti a funkcie aplikácie. Koniec kapitoly je venovaný popisu problémov spojených s implementáciou a popisu ich riešenia.

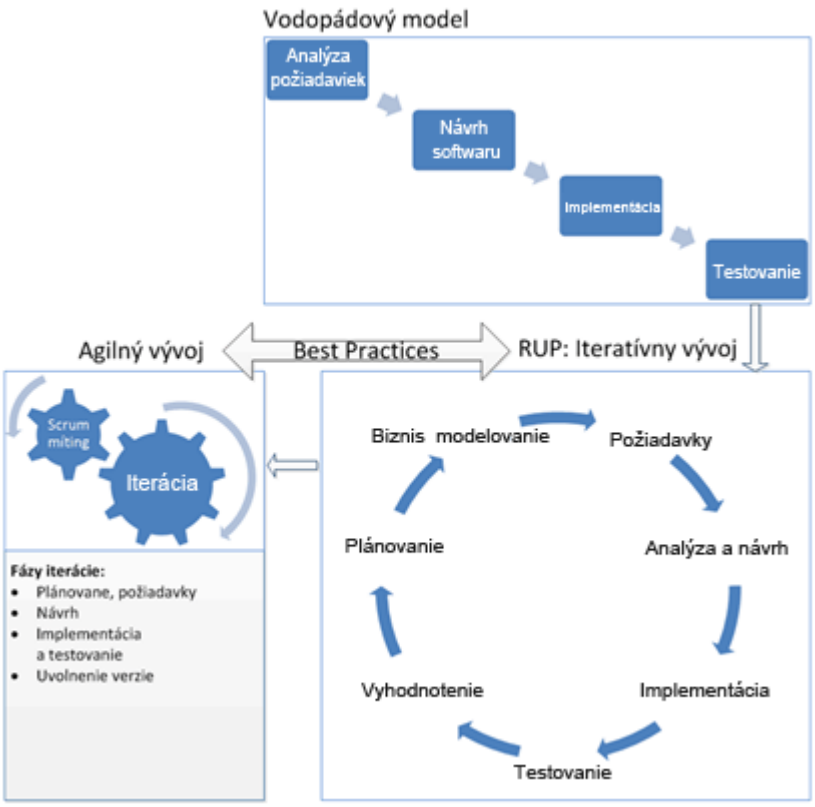
V šiestej kapitole je predstavená simulačná metóda Discrete-Event Simulation (DES) spolu s aplikáciou, ktorá bola implementovaná ako doplnková časť práce. Aplikácia využíva ku simulácii softwarového procesu práve metódu DES.

Na záver, v poslednej kapitole sú zhrnuté dosiahnuté výsledky, ich zhodnotenie a stručne predstavené plány do budúcna. Súčasťou poslednej kapitoly sú aj licenčné podmienky spojené s aplikáciou vyvinutou v rámci diplomovej práce.

2 Teoretická časť

2.1 Tradičné metódy vývoja

Vodopádový model, pozostáva zo všetkých štandardných vývojových fáz usporiadaných sekvenčne za sebou a ide o najstarší model životného cyklu vývoja softwaru. Vo veľkom bol využívaný ako pri menších, tak pri rozsiahlych projektoch. Jednou z výhod vodopádového modelu je napríklad jednoduchosť jeho implementácie. Uprednostňovaný býva pri projektoch kde sú na začiatku presne definované požiadavky, ktoré zostávajú pevne dané. Tam kde je kladený väčší dôraz na výslednú kvalitu pred dôrazom na dodržanie plánovaného času trvania alebo ceny projektu. Príkladom inštitúcií ktoré ešte dnes využívajú vodopádový model je napríklad spoločnosť NASA. Systémy pri ktorých je kladený vyšší dôraz najmä na kvalitu a bezpečnosť pred rozpočtom sú označované ako Mission-Critical Systems. Pokiaľ ale hovoríme o vývoji softwaru, keď dopredu nepoznáme všetky požiadavky alebo je pravdepodobné že aktuálne požiadavky môžu byť zmenené nie je vhodné použiť vodopádový model. Hlavným dôvodom toho, prečo nie je vodopádový model vhodný je práve jeho hlavná myšlienka. Tá spočíva v naplánovaní všetkého na začiatku vývoja a nepočíta sa so zmenami v neskorších fázach. Pokúšať sa rozhodnúť o všetkom na úplnom začiatku môže však znamenať v komerčnej sfére vývoja softwaru veľkú pravdepodobnosť zlyhania. Napríklad riziko, že výsledný produkt nebude akceptovaný zákazníkom, nedodržanie stanovených míľnikov projektu, alebo nedodržanie predpokladaného rozpočtu. [1]



Obrázok 1: Ukážka metód vývoja softwaru.

2.2 Agilné metodológie v komerčnej sfére vývoja softwaru [1]

Softwarový vývojári si začínajú uvedomovať, že tradičné metódy vývoja softvéru, ktoré sú založené na vodopádovom modely, v komerčnej sfére vývoja v malých a stredne veľkých spoločnostiach veľmi nefungujú. Niektorí vývojári majú tendenciu povedať, že vodopádový model je zlý, alebo nepoužiteľný. Takéto tvrdenie však nie je správne a vždy si treba uvedomiť, ktorý z modelov, alebo prístupov je vhodný pre konkrétny projekt. Objavuje sa dopyt po nových metodológiách vývoja, ktoré sú schopné vyriešiť problém zahrnutia zmeny do požiadaviek aj v priebehu vývoja, prípadne v jeho neskorších fázach. Práve agilné metódy vývoja softvéru založené na iteratívnom prístupe, poskytujú riešenie pre tento problém a teda sú schopné akceptovať zmeny aj v neskorších fázach.

2.3 Porovnanie agilných a tradičných metód

Tradičné prístupy boli používané pri vývoji pomerne dlhý čas. Vodopádový model bol rozsiahlo použitý ako pri veľkých, tak pri menších projektoch, ktoré boli úspešne dokončené. Avšak aj napriek úspechu obsahuje viaceré nedostatky, neflexibilnú reakciu na meniace sa požiadavky a vysoko formálne procesy bez ohľadu na rozsah projektu. Kent Back zobral na vedomie tieto nedostatky a predstavil Extreme Programming, prvú agilnú metodológiu. Agilné metódy rátajú s nestálymi a meniacimi sa požiadavkami využívaním množstva techník, ktoré sú zamerané na spolupráci medzi vývojármi a zákazníkmi. Rozdiely medzi agilnými a tradičnými metódami sú zhrnuté nižšie v tabuľke 1. [9]

Tabuľka 1: Rozdiely medzi agilnými a tradičnými metodológiami.

	Agilné metódy	**Tradičné metódy**
Prístup	adaptívny	prediktívny
Miera spokojnosti	obchodná hodnota	dodržanie plánu
Rozsah projektu	malé	veľké
Štýl riadenia	decentralizovaný	autokratický
Pohľad na zmenu	zmena prispôsobivosti	zmena udržateľnosti
Kultúra	vedenie - spolupráca	príkaz - kontrola
Dokumentácia	minimálna	rozsiahla
Dôraz na	ľudí	procesy
Cykly	vyšší počet	obmedzený počet
Doména	nepredvídateľná	predpovedaná
Plánovanie vopred	minimálne	rozsiahle
Návrat investícií	v skorých fázach	po dokončení projektu
Veľkosť tímov	malé	veľké

[1] Agilné metódy sú používané aj v iných oblastiach, napríklad výskumných. Avšak v tejto práci sú poňaté v súvislosti z komerčnou sférou vývoja softwaru.

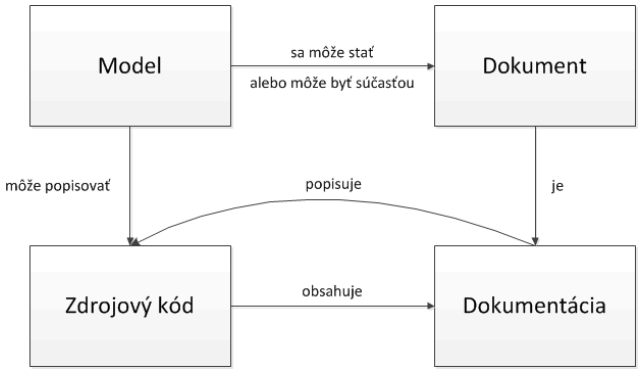
2.4 Vlastnosti a prehľad agilných metodológii

V tejto kapitole sú najskôr uvedené základné vlastnosti agilných prístupov a potom sú predstavené dve najpoužívanejšie agilné metódy, Extreme Programming a SCRUM.

Bolo potrebné nájsť modely ktoré umožňujú zmenu požiadaviek aj počas vývoja. Agilné metódy prinášajú v porovnaní s rigoróznymi metódami, ktoré sú založené na vodopádovom modely, odlišný pohľad na vývoj softvéru. Základná definícia v softwarovom inžinierstve hovorí, že cena zmeny exponenciálne rastie, čím bližšie je software ku fáze pripravenia k používaniu. Práve preto tu bola snaha identifikovať a implementovať zmeny čo najskôr. Avšak, pri vhodnej kombinácií technológií a programátorských praktík je možné takýto rast ceny zmien zastaviť. Pri tradičných metódach sa cena odvíja od nutnosti znovu zopakovať celý vývojový cyklus. Vykonanie zmien sa v ňom nachádza vo fáze údržby. Vývojár musí zmeniť požiadavky, prerobiť špecifikáciu a návrh, implementovať zmenu a nakoniec znovu otestovať celú aplikáciu aby mohol zabezpečiť to že všetko funguje rovnako správne ako pred vykonaním zmeny. Navyše cena sa odvíja aj od rozsahu požadovanej údržby, množstva dokumentov a zabezpečenia drahých testov. Implementovanie zmeny nebýva jednoduché, pokiaľ sa pri vývoji architektúry nepočítalo s prípadnými zmenami. Agilné metódy reagujú na tieto požiadavky a podporujú jednoduchšie aplikovanie zmien. Nevyžadujú veľké množstvo dokumentácie a viac podporujú automatizované testovanie pomocou unit testov . [1]

2.4.1 Dokumentácia pri agilnom vývoji

Agilný vývoj nabáda ku tvorbe iba čo najnutnejšej a dostačujúcej dokumentácie. Pri agilnom vývoji je dokumentácia považovaná za nutné zlo a dôležitejšia ako dokumentácia je komunikácia. Dokumentácia je vytvorená vtedy, pokiaľ ide o najlepší spôsob ako dosiahnuť stanovené ciele. Často sú ale pre dosiahnutie cieľov dostupné lepšie spôsoby ako písanie dokumentácie. Lepšie vynaložený čas je v tomto prípade na testovanie produktu zákazníkmi alebo vývojármi, ako na písanie veľkého množstva dokumentácie. Niekedy je však nutné vytvoriť dokumentáciu pre lepšiu podporu komunikácie s externými vývojármi alebo iba preto lebo to vyžaduje zákazník. Dôležité zabezpečiť to, aby mala dokumentácia väčší prínos v porovnaní s cenou jej vytvorenia a udržiavania. Preto si treba uvedomiť či ju iba chceme, alebo skutočne potrebujeme. Aktualizovať dokumentáciu je vhodné iba vtedy pokiaľ je to skutočne potrebné. [1], [2]



Obrázok 2: Vzťah medzi modelmi, dokumentmi, zdrojovým kódom a dokumentáciou.[2]

2.4.2 Jednoduchý návrh softwaru

Agilné metódy uprednostňujú jednoduchý návrh z dôvodu zabezpečenia čo najjednoduchšieho uplatňovania prípadných zmien v kóde. Podstatné je vymyslieť jednoduché riešenie, ktoré je možné uplatniť pre konkrétnu vlastnosť. Pokiaľ riešenie návrhu nie je dostatočne dobré, je pre splnenie požiadaviek použitý refraktoring [2]. Refraktoring musí byť taktiež použitý na udržanie jednoduchej štruktúry softvéru a na odstránenie duplicitného kódu. Podstatou návrhu počas refraktoringu je udržať čistý kód a umožniť vývojárom navrhnúť iba to čo je požadované pre implementovanie ďalšej funkcie. Vývojári môžu byť neochotní zabezpečiť požadovaný refraktoring pre udržanie jednoduchého kódu, v prípade nedodania dostatočne rozsiahlych testovacích prípadov. [1]

2.4.3 Testovanie a kvalita softwaru

Dôležitou súčasťou pri využívaní agilných metodológií je aj testovanie. V porovnaní s rigoróznymi metódami, kde je testovanie umiestnené na konci vývojového cyklu, veľa agilných metód uprednostňuje Test-Driven Developement (TDD), teda vývoj riadený testovaním. Pri vývoji riadenom testovaním je nutné písať testy ešte pred napísaním samotného zdrojového kódu. Tieto testy sú písané ako automatizované unit testy, ktoré by vždy mali bežať úspešne, s výnimkou obdobia počas ktorého dochádza k implementácií nových funkcií alebo oprave chýb. Tieto testy poskytujú rýchlu odozvu na zmeny. Metódy extrémneho programovania (XP) uvádzajú že unit testy a funkčné akceptačné testy sú prospešné pre projekt. Kým tieto testy nie sú nevyhnutne tak dobré ako testy vytvorené profesionálnymi testermi, organizácie prijímajúce XP zistili, že s ich zavedením došlo k zvýšeniu kvality sofwaru. Vývoj riadený testovaním je jednou z nevyhnutných praktík XP, ktoré by mali byť uplatnené vždy, keď dochádza ku postupnému prijímaniu praktík. Pri vývoji riadenom testovaním je zaznamenaný výrazný pokles chýb pri písaní zdrojového kódu. [4]

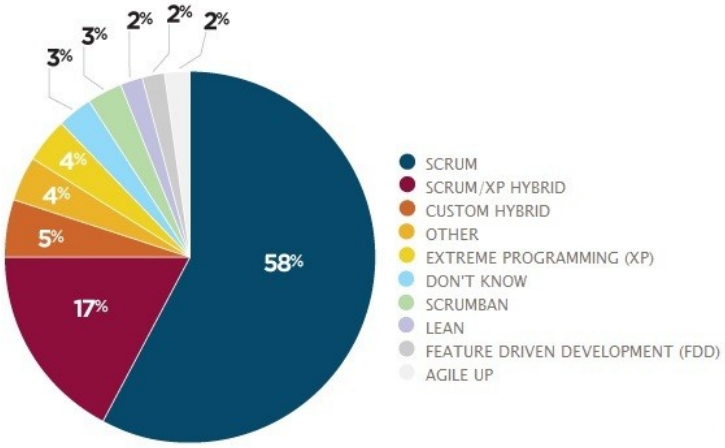
Vzhľadom na to, že návrh je jednoduchý, neplánovaný na úplnom začiatku a dokumentácia je menej rozsiahla, pre zástancov tradičného vodopádového modelu môže byť otázna kvalita vytvoreného kódu. Agilné metódy však obsahujú atribúty kvality. Ako už bolo spomenuté, vývoj riadený testovaním (TDD) viditeľne redukuje množstvo jednoduchých chýb a jednoduchý dizajn, ktorý je často menený redukuje časti kódu, ktoré nie sú nevyhnutné a udržiava vyššiu kvalitu kódu. Agilné metódy taktiež zahŕňajú kontinuálnu integráciu, ktorá zabezpečuje to, že hneď ako je to možné sú časti kódu integrované do hlavného kódu. Toto redukuje chyby, ktoré sa objavujú v prípadoch keď ku integrácii dochádza iba zriedkavejšie. Navyše, môžu byť zautomatizované aj akceptačné testy. Na záver, s využívaním agilných metód dochádza ku častejšej kontrole kvality ako je to pri rigoróznych metódach vývoja softwaru. [1], [4]

[2] Refraktoring znamená zmenu aktuálnej internej štruktúry kódu, bez vplyvu zmeny na vonkajšie správanie. [1],[2]

2.5 Aliancia agilného vývoja a miera používania agilných metodík

V roku 2001 vznikla skupina tvorená sedemnástimi členmi a nazvaná The Agile Alliance. Vytvorili manifest, ktorý stanovuje základy pre agilný vývoj softvéru. Nižšie sú uvedené hodnoty agilného manifestu. Položky na pravej strane sú síce v manifeste braté na vedomie, no dôležitejšie sú položky uvedené na ľavej strane. [3]

- **Jednotlivci a interakcia** sú dôležitejšie ako procesy a nástroje
- **Fungujúci softvér** je dôležitejší ako rozsiahla dokumentácia
- **Spolupráca zákazníka** je dôležitejšia ako rokovanie o zmluve
  Reakcia na zmeny je dôležitejšia ako dodržiavaním plánu



Obrázok 3: Miera používania agilných metód vývoja za rok 2010. [21]

2.6 Xtreme Programming

Technika Extreme programming (XP) vznikla kvôli problémom, ktoré boli spôsobené dlhými vývojovými cyklami tradičných vývojárskych modelov. XP proces zahŕňa krátke vývojové cykly, inkrementálne plánovanie, priebežnú odozvu a spolieha sa predovšetkým na komunikáciu. Všetky tieto metódy umožňujú XP programátorom uplatňovanie zmien s oveľa väčšou odvahou. Členovia XP tímu trávia niekoľko minút denne programovaním, niekoľko minút riadením projektu, niekoľko minút návrhom, niekoľko minút zabezpečením odozvy a každý deň niekoľko minút teambuildingom. Zhrnutie základných princípov ktoré je nutné v XP dodržiavať:

- **Plánovanie** - Programátor odhadne úsilie potrebné pre implementáciu stories a zákazník rozhodne o rozsahu a načasovaní releases.
- **Malé/krátke releases**[3]**:** Aplikácia je vyvíjaná v skupinke malých a často aktualizovaných verzií. Nové verzie sú uvoľňované v denných až mesačných intervaloch.
- **Metafora** - Systém je definovaný metaforami medzi zákazníkom a programátormi, z dôvodu zlepšenia komunikácie. Popisujú ako systém funguje.

[3] Anglické slovo release, ktoré v preklade znamená uvoľnenie (vo význame napr. uvoľnenie verzie).

- **Jednoduchý návrh -** Dôraz je kladený na návrh najjednoduchšieho možného riešenia, ktoré je implementované a okamžité odstránenie nadbytočného kódu.
- **Refraktoring** - Umožňuje prestavbu systému odstránením duplikácií, zlepšením komunikácie, zjednodušením a zvýšením flexibility a to bez meniacej sa funkcionality programu.
- **Programovanie v páre** - Všetok kód je napísaný dvoma programátormi pri jednom počítači.
- **Kolektívne vlastníctvo** – Za zdrojový kód je zodpovedný každý člen tímu.
- **Priebežná integrácia -** Nové časti kódu sú integrované do systému tak rýchlo ako je to možné. Po integrácii je systém znovu zostavený a otestovaný. Všetky testy musia úspešne prejsť aj po vykonaní zmien.
- **40-hodinový týždeň -** Nikto nemôže pracovať viac ako 40 hodín za týždeň.
- **Na strane zákazníka -** Zákazník musí byť kedykoľvek dostupný vývojárom.
- **Štandardy v písaní kódu -** Existujú pravidlá, ktoré sú dodržiavané programátormi, tak aby to prinieslo celistvosť kódu a uľahčilo komunikáciu medzi vývojármi.

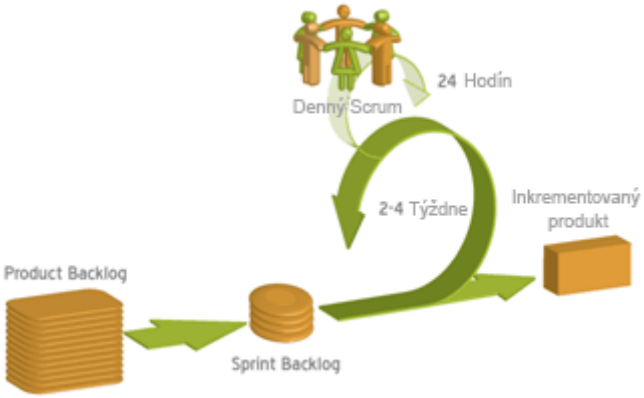
Životný cyklus XP projektu je rozdelený do 6 fáz: Prieskum, Plánovanie, Iterácie na uvoľnenie, Produkcia, Udržateľnosť a Smrť. Vo fáze prieskumu zákazník napíše na kartičky čo si želá, aby jeho program spĺňal. Toto vedie ku fáze plánovania, kde je každému užívateľskému príbehu priradená priorita a taktiež tu je naplánované prvá Release. Prvá iterácia vývojárskeho tímu slúži na vytvorenie architektúry celého systému. Potom priebežne integrovať a testovať kód. Prídavné testovanie a kontrola výkonu systému pred tým ako bude uvoľnený zákazníkovi je vykonaná v produkčnej fáze. Odložené nápady a pripomienky počas tejto fázy sú zdokumentované kvôli neskoršej implementácií v aktualizovanej Release, ktorá je vytvorené vo fáze údržby. Nakoniec je tu fáza smrti, ktorá je na rade vtedy, keď zákazník nemá žiadne nové požiadavky pre implementáciu. Taktiež bola napísaná všetka nevyhnutná dokumentácia k systému a nie sú potrebné žiadne ďalšie zmeny v architektúre, návrhu, alebo zdrojovom kóde. [1],[9]

2.7 SCRUM

Scrum je iteratívny, jednoduchý a prispôsobivý framework pre riadenie vývoja softwarových projektov v menších tímoch, ktorý umožňuje výber vhodných prvkov pre návrh softwarového procesu. SCRUM vo fáze implementácie nedefinuje žiadne špecifické metódy. Zameriava sa na správnosť vývoja systému podľa požiadaviek a to aj v meniacom sa podnikateľskom prostredí.

Vývoj softvéru zahŕňa niekoľko premenných ktoré sa počas práce na projekte často menia. Sú to požiadavky, zdroje, časový rámec a technológie. Namiesto toho, aby sa skúšalo niečo veľmi detailne naplánovať, Scrum definuje empirické riadenie procesu. Dôvodom je zabezpečenie viditeľnosti aktuálneho stavu projektu, aby sa predišlo vyvolaniu chaosu kvôli nepredvídaným udalostiam. Vyžaduje to určité manažérske praktiky a nástroje v odlišných fázach vývoja.

Keďže Scrum je iteratívny proces, projekt vyvíjaný za pomoci Scrumu je rozdelený do šprintov, ktoré sú iteráciami Scrumu. Je taktiež inkrementálny, to znamená, že každý šprint vyprodukuje spustiteľnú verziu softvéru s prídavnou funkcionalitou. [21]



Obrázok 4: Scrum Framework [20]

V nasledujúcich podkapitolách sú uvedené role a artefakty v anglickom jazyku. Tieto termíny by bolo možné preložiť ale vzhľadom na zavedené používanie týchto termínov v súvislosti zo Scrumom zostaneme pri anglických názvoch. Zoznam rolí, artefaktov a ich definície je umiestnený v prílohe C, na strane IV.

2.7.1 Role

Scrum definuje pre členov projektu tri základné role.

- **Product Owner –** Pozerá sa na projekt z pohľadu podnikateľa a zadáva požiadavky na výsledný produkt a plánuje releases. Product Owner je zodpovedný aj za určovanie priority požiadaviek, počas celého vývoja projektu.
- **Team –** Členovia tímu sú zodpovední za vývoj produktu v jednotlivých šprintoch. Tím je samo-organizačná jednotka, ktorá je zodpovedná za vývoj požadovanej funkcionality na konci iterácie. Tím si sám vyberá metódy použité pri vývoji a jeho členovia si sami vyberajú úlohy, na ktorých chcú pracovať. Optimálna veľkosť tímu pre SCRUM je okolo 6 členov. Neodporúča sa, aby počet členov bol viac ako 10. Jeden z podstatných dôvodov, je aj to, že menšie tímy pracujú efektívnejšie ako väčšie.
- **Scrum Master –** Túto rolu zastáva človek, ktorý je niečo ako projektový manažér v Scrume. V kontraste s klasickým projektovým manažérom, Scrum Master nevedie tím, ale má za úlohu motivovať členov tímu a pomáhať im prekonávať prekážky pri vývoji. Má zodpovednosť za dodržiavanie Scrum procesu a jeho úlohou je aj naučiť jednotlivých členov tímu ako funguje Scrum proces.

2.7.2 Praktiky

- **Product Backlog** – Zoznam všetkých vlastností a zmien, ktoré sa týkajú systému, usporiadaných podľa priority. Product Backlog nieje nikdy hotový v priebehu vývoja projektu, ale jeho položky sa neustále menia s meniacimi sa požiadavkami na produkt. Pokiaľ sú splnené všetky jeho položky, projekt sa považuje za dokončený. Zodpovedný za údržbu tohto zoznamu je Product Owner.

- **Effort Estimation –** Odhad úsilia je v Scrume iteratívny proces. Najskôr sa zaznamenajú počiatočné odhady a neskôr sa tieto odhady môžu ešte meniť, respektíve upresňovať. Je tak potom možné vidieť históriu jednotlivých odhadov.
- **Sprint –** Podstatou Scrumu sú práve šprinty. Jedná sa o 30-dňové iterácie, počet dní však nie je presne stanovený a ich dĺžka môže byť od jedného do štyroch týždňov. Každý šprint prináša novú funkcionalitu. Počas šprintu členovia tímu vyberajú čo najvhodnejšie metódy na dosiahnutie cieľa, s podporou Scrum Mastera.
- **Sprint Planning meeting –** Pred začiatkom každého šprintu je zvolávaný dvojfázový míting. V prvej fáze mítingu sa rozhodne o cieľoch a prioritách. V ďalšom šprinte, Product Owner vyberie z Product Backlogu tie najdôležitejšie vlastnosti a členovia tímu ho informujú o tom, koľko z nich stihne počas šprintu implementovať. V druhej fáze mítingu sú vytvárané úlohy pre Sprint Backlog.
- **Sprint Backlog –** Štartovací bod pre každý šprint. Na začiatku sú vybraté určité položky z Product Backlogu a tie sú neskôr upravované. Nie sú tu však pridávané nové požiadavky, tie môžu byť pridané iba do Product Backlogu.
- **Daily Scrum meeting –** Denné mítingy slúžia na to, aby bol každý člen tímu v obraze. Ide o krátke mítingy s presne definovanými témami. Na každom mítingu odpovedá vývojár na 3 otázky. *Čo urobil od posledného mítingu? Čo plánuje robiť do ďalšieho mítingu? Stretol sa s nejakými prekážkami?* Vývojári sa často stretávajú s podobnými, alebo rovnakými problémami, o ktorých sa navzájom dozvedia práve na mítingu.

Scrum bol úspešne použitý pri tisíckach projektoch v organizáciách, ktoré zaznamenali podstatné zlepšenie vo vývoji. Scrum je vhodný pre menšie tými pracujúce na menších projektoch. Pri vývoji rozsiahlejších projektov je vhodné aby oddelené menšie tými využívali niektoré prvky Scrumu. Vzhľadom na to, že Scrum nie je dogma a nie je nutné pri jeho používaní dodržiavať všetky pravidlá, často sa kombinuje spolu s inými praktikami. Dobrá kombinácia je použitie XP praktík so Scrumom. Je však veľmi dôležitý výber vhodných prvkov a nesmú byť porušené základné praktiky. Porušenie kľúčových praktík a nepochopenie rolí v Scrume vedie k neúspechu. [1], [9], [20], [21]

2.8 Existujúce on-line nástroje pre podporu agilných procesov

Táto podkapitola predstavuje tri on-line aplikácie pre podporu procesov založených na agilných metodológiách, prevažne však na Scrume. Jedná sa o pomerne rozdielne aplikácie a každá z nich ma svoje špecifické výhody v podobe funkcií, ktorými disponuje. Popíšeme jednotlivé riešenia z pohľadu jej prevádzkovateľa, potom zhrnieme kľúčové vlastnosti a taktiež sa pokúsime o popis ich nedostatkov.

2.8.1 SeeNowDo

Jednoduchý a flexibilný nástroj, ktorý slúži na zdieľanie informácií o projekte a jednotlivých úlohách, buď to v rámci tímu, alebo v rámci niekoľkých tímov. SeeNowDo je zdarma. Umožňuje spoluprácu na

niekoľkých projektoch a užívateľské rozhranie je veľmi intuitívne. Na obrázku 5 môžeme vidieť ukážku tabule s úlohami, z prostredia SeeNowDo. [17]

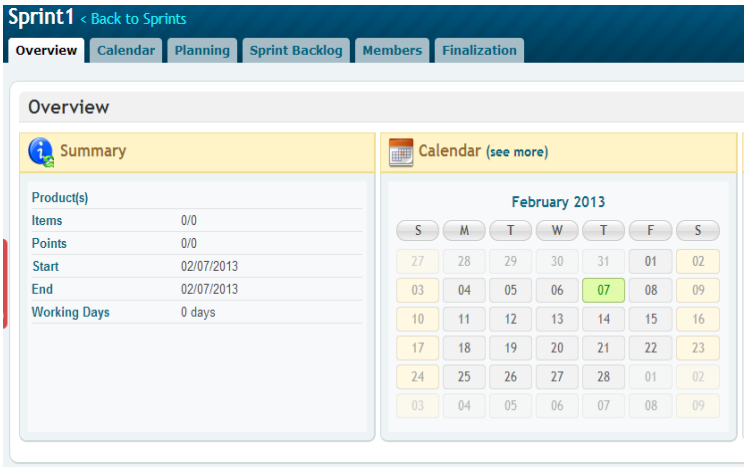
Obrázok 5: Ukážka aplikácie SeeNowDo.

Výhodou tejto aplikácie je skutočne jej intuitívne grafické rozhranie. Projekty sú tu rozdelené na iterácie, ktoré po otvorení obsahujú jednotlivé stories a úlohy. Ako projektom, tak aj iteráciám je možné priradenie stavu, v ktorom sa práve nachádza. Taktiež je tu možnosť prispôsobenia si tabule s úlohami a to vytváraním vlastných stĺpčekov, do ktorých sú následne pridávané úlohy. Úlohy je možné kategorizovať a farebne odlíšiť podľa ich typu. Tak je zabezpečená veľmi dobrá prehľadnosť.

Riešenie je naprogramované za pomoci technológie Silverlight od Microsoftu a to je prekážkou pri zobrazovaní na Apple zariadeniach. SeeNowDo nie je optimalizované pre mobilné zariadenia. Chýba tu evidencia členov, prípadne tímov pracujúcich na projekte.

2.8.2 **PangoScrum**

Ďalší on-line nástroj pre Scrum, ktorý je k dispozícii zdarma. Jedná sa o aplikáciu jednoduchú na používanie s taktiež pomerne intuitívnym prostredím. Ukážku z prostredia aplikácie môžeme vidieť na nasledujúcom obrázku. [18]



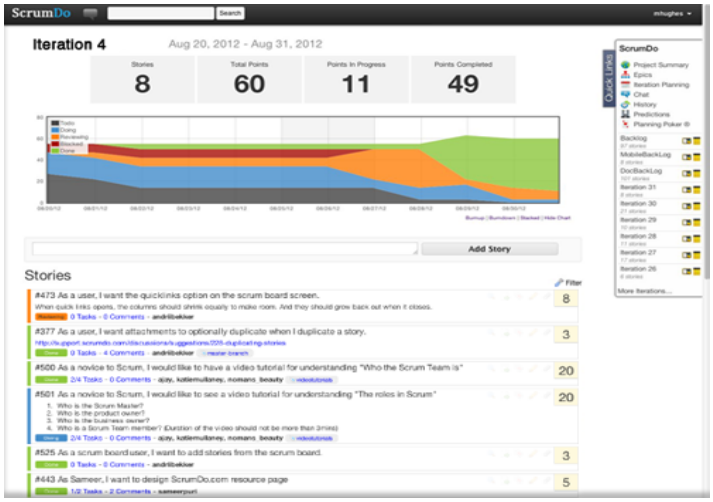
Obrázok 6: Ukážka aplikácie PangoScrum.

Výhodou aplikácie je presná evidencia Product Backlogu a Sprint Baclogu pri jednotlivých projektoch. Obsahuje evidenciu členov tímov a počíta s rolou Scrum Mastera. Plánovací kalendár umožňuje napríklad naplánovanie plánovacích mítingov, denných mítingov. Vlastnosti aplikácie pomerne presne kopírujú základné vlastnosti Scrumu.

Neobsahuje však tabuľu s úlohami, na ktorej by bolo možné prehľadné zobrazenie úloh, čo je pomerne výrazný nedostatok. Grafické rozhranie nie je optimalizované pre mobilné zariadenia.

2.8.3 ScrumDo

Z troch popisovaných nástrojov sa jedná o najprepracovanejší nástroj. Obsahuje množstvo funkcií, ktoré sú dôležité pre podporu Scrumu, ale taktiež obsahuje rozširujúce funkcie. Napríklad hru plánovací poker, alebo možnosť predikcie ďalšieho priebehu. Aplikácia však na rozdiel od dvoch predošlých nie je zdarma. [19]



Obrázok 7: Ukážka aplikácie ScrumDo.

Výhodou aplikácie je pestrý výber funkcií, ktoré sú vymyslené práve pre podporu Scrumu. Veľmi silnou stránkou tejto aplikácie je však možnosť efektívneho plánovania v meniacom sa prostredí a zároveň aj predikciu vývoja. Obsahuje prehľadnú tabulu s úlohami, kde je možné vidieť úlohy na ktorých sa práve pracuje a ich aktuálny stav na prehľadnej tabuli.

Nevýhodou aplikácie je jej cena, ktorá sa pohybuje v závislosti na funkciách, ktoré chce zákazník využívať a taktiež v závislosti na počte vývojárov a veľkosti diskového priestoru.

3 Rozhodujúce faktory úspechu softwarových projektov

Sledovať pokrok a robiť jednoduché rozhodnutia sú dva nevyhnutné faktory pre úspešne riadenie projektov. Práve softwarové nástroje by mali tieto dva faktory podporovať. Riadenie projektov je však oveľa zložitejšie. Existuje veľa nástrojov pre podporu vývoja softwaru. Niektoré nástroje slúžia všeobecne pre podporu softwarových procesov, ale existujú aj úzko špecializované nástroje práve pre SCRUM. Takýto nástroj môže zabezpečiť úspech v prípade, že spĺňa požiadavky Scrum tímu. [5]

3.1 Úloha nástrojov pre podporu vývoja

Podporou vývoja softwaru je myslená automatizácia niektorých úkonov, napríklad zlepšenie plánovania za pomoci predikčných a simulačných metód. Za zlepšenie plánovania môžeme považovať napríklad odhad vynaložených zdrojov na vývoj projektu, alebo zloženie vývojárskych tímov.

3.1.1 Nepotrebujeme žiadne nástroje

Scrum nevyžaduje pre svoju podporu žiadne softwarové nástroje. Pri niektorých projektoch sa využíva ako pomôcka klasická tabuľa, na ktorú sa pripínajú lístočky s úlohami. Denné stretnutia pred klasickou tabuľou umožňujú veľmi jednoduchú manipuláciu s úlohami a priamočiaru komunikáciu členov tímu. Tabuľa je však riešenie, ktoré má aj svoje nevýhody. Jednou z nich môže byť napríklad rušivý vplyv, pokiaľ ľudia vo veľkom open-office chodia neustále k tabuli a rušia tak ostatných. Úlohy zaznamenané na tabuli môže niekto zmeniť bez toho aby mal na to oprávnenie a takáto zmena ani nebude zaregistrovaná.

3.1.2 Výhody používania softwarových nástrojov

Softwarové nástroje môžu pomáhať projektovému manažmentu pri odhade zdrojov ktoré budú vynaložené na vývoj. Môže sa jednať o ľudské alebo finančné zdroje. Taktiež je možné odhadnúť čas trvania práce na projekte. Takýmto odhadom môže pomáhať napríklad simulácia softwarového procesu, ktorú umožňujú sofistikovanejšie softwarové nástroje. Softwarová podpora je nevyhnutná v prípade, že na projekte pracuje viacero tímov z iných miest a nemajú možnosť častejších a pravidelných stretnutí.

3.2 Ponaučenie

Aj napriek tomu, že softwarový proces založený na frameworku SCRUM síce nevyžaduje podporu softwarových nástrojov, je takýto typ podpory veľmi užitočný. Pomáha projektovému manažmentu a vývojárom k uľahčeniu a efektívnejšiemu vykonávaniu svojej práce. Softwarový nástroj pre podporu môže v prípade distribuovaných tímov slúžiť aj ako komunikačný prostriedok.

4 Návrh riešenia

4.1 Špecifikácia zadania

Cieľom práce je implementovať aplikáciu pre podporu softwarového procesu v komerčných spoločnostiach. Biznis procesy sú špecifikované všeobecne. V popise procesov je zachytený navrhnutý priebeh Scrum procesu podporovaného aplikáciou pre projektový manažment. Zoznam systémových a procesných rolí sa nachádza v prílohe C na strane II.

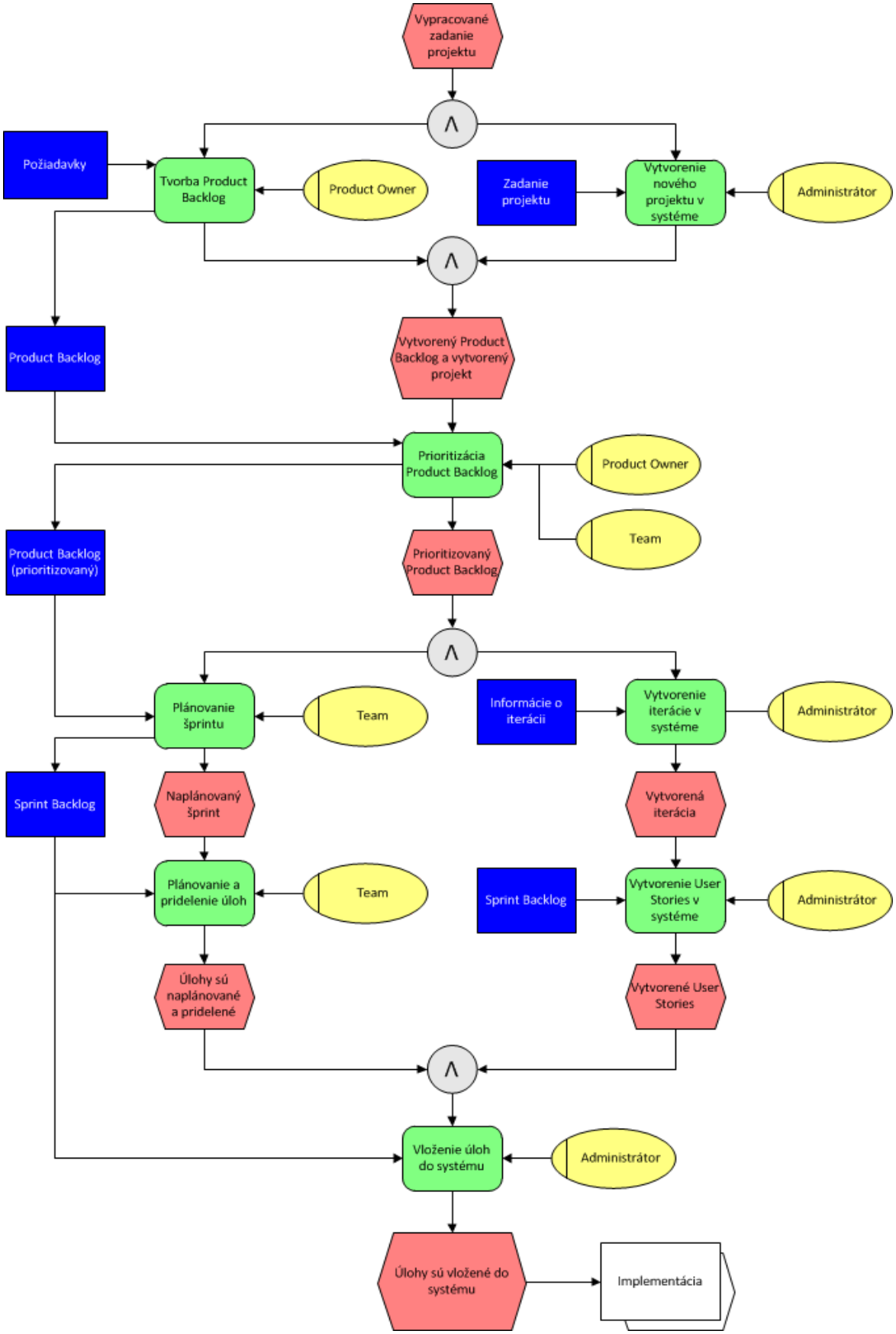
4.1.1 Špecifikácia procesu plánovania v Scrume

Prvý navrhnutý proces popisuje fázu plánovania vývoja a zadávanie konkrétnych úloh do systému. Proces zachytáva fázu plánovania celého projektu, zostavenia dokumentu Product Backlog, následne aj plánovanie prvej iterácie spolu ktorého výstupom je Sprint Backlog. Taktiež je tu zachytená práca s aplikáciou, kde sú vložené informácie o novom projekte a taktiež informácie o prvej iterácii a konkrétnych úlohách pre vývojárov.

Po vypracovaní zadania má Product Owner podľa zozbieraných požiadaviek za úlohu vytvoriť dokument Product Backlog. V rovnakej fáze by mal administrátor v systéme vytvoriť podľa zadania nový projekt. Po vytvorení Product Backlogu následne Product Owner nastaví prioritu jednotlivým položkám dokumentu. Prioritizácia prebieha za účasti členov tímu, ktorí sa podieľajú na vývoji projektu. Prioritizovaný Product Backlog je vstupný dokument pre plánovanie šprintu.

Plánovanie šprintu prebieha v dvoch fázach na plánovacom mítingu. V prvej fáze Product Owner prezentuje pred tímom Product Backlog, ktorý vytvoril na základe požiadaviek. Tím po pochopení obsahu jednotlivých položiek prezentovaného dokumentu rozhoduje o jeho odsúhlasení. Počas druhej fázy má tím za úlohu naplánovať šprint. Po vytvorení Product Backlogu a prioritizovaní jeho položiek je proces rozdelený do dvoch paralelných vetiev. V prvej vetve prebieha plánovanie šprintu a plánovanie a rozdelenie jednotlivých úloh. Toto plánovanie má na starostí tím, pretože tím je zodpovedný za riadenie vlastnej práce. V druhej vetve prebieha vytváranie položiek v systéme.

Vstupným dokumentom pre plánovanie šprintu je prioritizovaý Product Backlog, ktorý je odsúhlasený členmi tímu. Výstupom plánovacieho mítingu je Sprint Backlog ktorý obsahuje položky vybraté z Product Backlogu a na ktorých sa bude pracovať v aktuálnom šprinte. Súčasne prebieha časť procesu v ktorej administrátor vytvorí novú iteráciu a vkladá do systému položky Sprint Backlogu, ktoré sú v systéme označené ako User Stories a vkladané do iterácie projektu. Iteráciou projektu v systéme je označený šprint. Jednotlivé položky Sprint Backlogu sú rozdeľované na menšie úlohy, na ktorých budú pracovať konkrétni členovia tímu. Rozdeľovanie a prideľovanie úloh má na starosti tím. Úlohy následne vkladá do systému administrátor, po tom ako sú rozdelené a taktiež po tom ako sú v systéme vytvorené potrebné User Stories. Každá úloha jej priradená konkrétnemu User Story, pod ktorý spadá. Udalosť ktorá hovorí že úlohy sú vložené do systému označuje koniec procesu plánovania. Popísaný proces je grafický znázornený na obrázku 8 pomocou EPC (Event-driven Process Chain) diagramu.



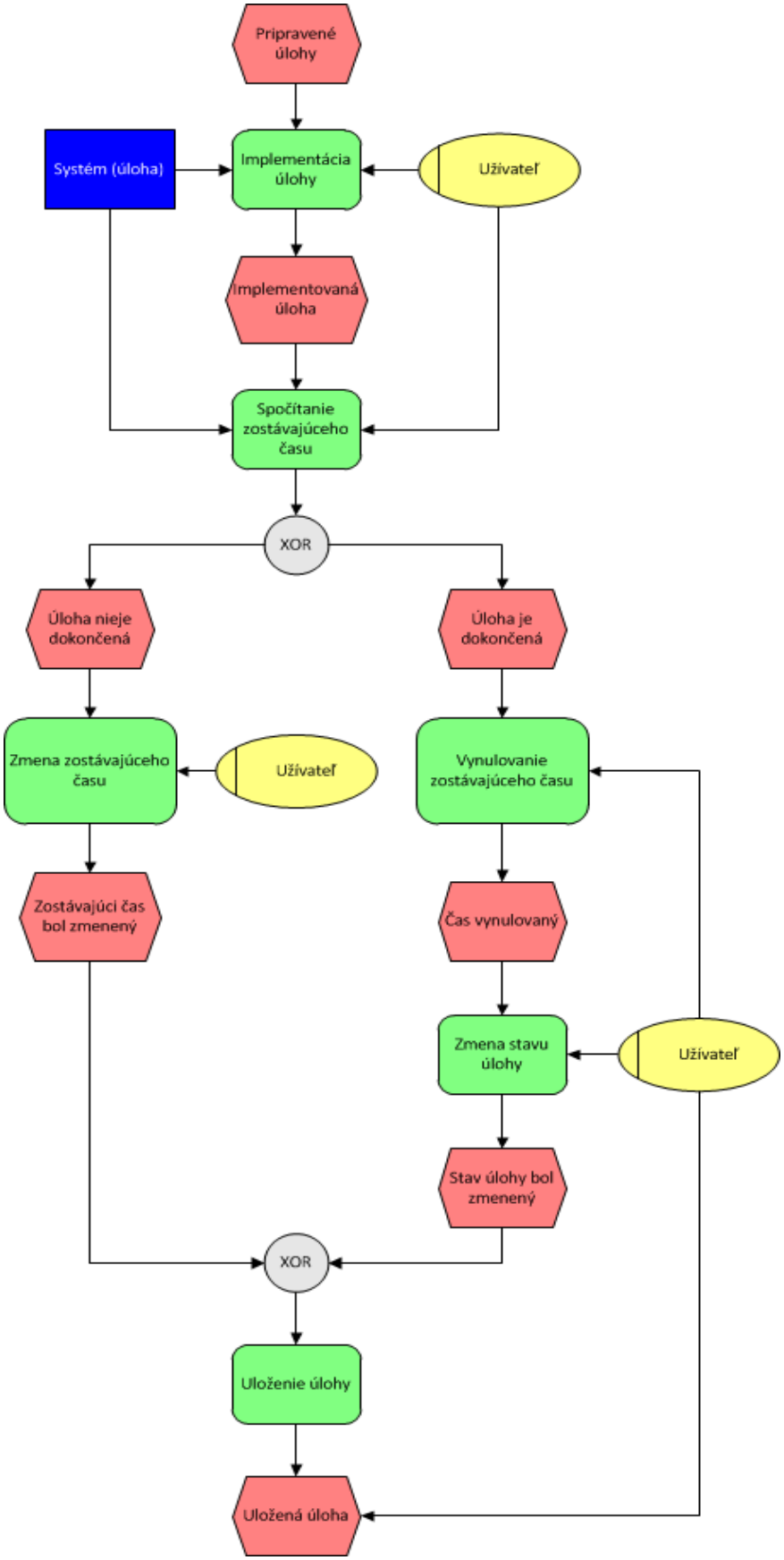
Obrázok 8: EPC diagram procesu Plánovania.

4.1.2 Špecifikácia procesu implementácie v Scrume

Požiadavkou je, aby po vytvorení úloh bola aplikácia pripravená pre vývojárov, ktorí majú možnosť meniť stav jednotlivých priradených úloh. Každá úloha je priradená vývojárovi, ktorý je zodpovedný za jej riešenie a taktiež je možné nastaviť viditeľnosť úlohy pre ostatných členov vývojárskeho tímu.

EPC diagram na obrázku 9 znázorňuje proces implementácie. Proces zachytáva implementáciu jednej konkrétnej úlohy v spolupráci s používaním aplikácie pre projektový manažment, kde užívatelia majú možnosť zobrazenia im priradených úloh a zmeny informácií o týchto úlohách. Predpokladom pre začiatok procesu sú pripravené úlohy, ako výstup procesu plánovania. Užívateľ (vývojár) implementuje jednotlivé úlohy o ktorých informácie čerpá zo systému. Po implementácii konkrétnej úlohy si spočíta čas, ktorý strávil jej implementáciou. Pokiaľ je zostávajúci čas väčší ako nula, úloha nie je dokončená a proces pokračuje zmenou zostávajúceho času a uložením úlohy. V prípade, že je počet zostávajúcich hodín nula a teda úloha sa považuje za dokončenú, vývojár nastaví hodnotu zostávajúceho času na nulu. Po zmene zostávajúceho času zmení stav úlohy. Stav úlohy môže zmeniť napríklad na *„dokončená“*. Zmena stavu úlohy zabezpečí jej premiestnenie do zvoleného stĺpčeka na tabuly. V prípade, že bol stav úlohy zmenený nasleduje uloženie úlohy. Udalosť uloženia úlohy označuje koniec procesu implementácie.

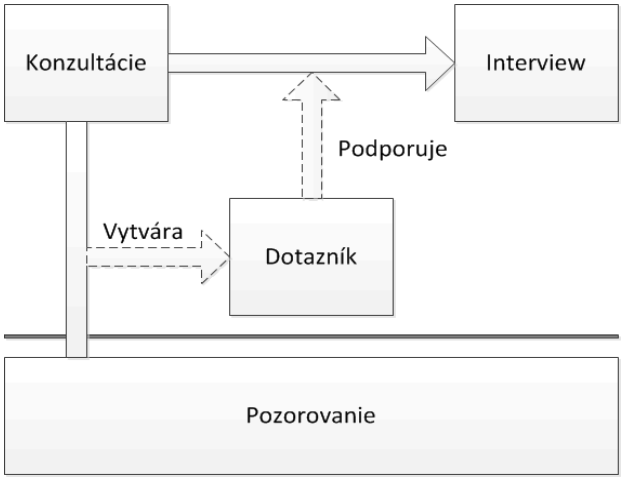
Vďaka možnosti zmeniť stav jednotlivých úloh, ich presunutím do vytvorených stĺpčekov je možné sledovať to, na ktorých úlohách sa aktuálne pracuje, ktoré úlohy sú dokončené, prípadne ktoré úlohy čakajú na overenie. Konkrétne stavy úloh závisia od prispôsobenia aplikácie pre špecifické potreby spoločnosti, alebo pre potreby riešeného projektu. Doporučené názvy stĺpcov sú napríklad: *‚nezačaté‘, ‚vo vývoji‘, ‚čakajúce na overenie‘ a ‚dokončenie‘*. Štruktúra stĺpcov pre jeden šprint je vždy rovnaká. V rôznych šprintoch však môže byť rôzna štruktúra stĺpcov.



Obrázok 9: EPC diagram procesu Implementácie.

4.2 Špecifikácia požiadaviek

Vzhľadom na to, že všetky navrhované procesy sú založené na frameworku SCRUM, obsahujú podobné prvky. V niektorých vlastnostiach sa však odlišujú a preto je potrebné prispôsobiť aplikáciu pre podporu procesu, presne podľa potrieb konkrétnej spoločnosti, ktorá ju bude používať. Nasledujúce podkapitoly obsahujú stručný popis spoločností. Ďalej popis požiadaviek pre návrh softwarového procesu a taktiež pre analýzu a návrh aplikácie. Požiadavky boli zozbierané pomocou konzultácií so zástupcami spoločností a taktiež pomocou vyplnených dotazníkov.



Obrázok 10: Ukážka modelu zberu požiadaviek pre modelovanie softwarových procesov.

4.2.1 Spoločnosť STAPRO

Spoločnosť STAPRO, s.r.o. je významným dodávateľom informačných systémov, diagnostických prístrojov, zdravotníckej techniky a zároveň aj poskytovateľom služieb v oblasti informačných technológií pre zdravotníctvo. Stapro od svojho založenia v roku 1990 priebežne zvyšuje tržný podiel a obrat predaja vlastného aplikačného softwaru a služieb. Dcérska spoločnosť STAPRO Slovensko, s.r.o. taktiež úspešne pôsobí na slovenskom zdravotníckom trhu už od roku 1993. [23]

4.2.2 Spoločnosť SCOVECO

SCOVECO, s.r.o. je skúsená dynamická spoločnosť. Bola založená v roku 2011 ako firma, ktorá si kladie za cieľ presadzovať jednoduché a inovatívne riešenia jednoduchých aj komplexných problémov pomocou moderných IT technológií. Zámerom spoločnosti je budovať, presadzovať a nasadzovať IT riešenia, ktoré uľahčujú život firmám a ľuďom. [24]

4.2.3 Skupina M7 a satelitná platforma Skylink

Skupina M7 bola založená v októbri 2009 v Luxemburgu a je európskym poskytovateľom satelitných služieb a zastrešuje: CanalDigitaal v Holandsku, TV Vlaanderen vo Flámsku and TéléSAT vo Francúzskej časti Belgicka, AustriaSat v Rakúsku a Skylink pre český a slovenský trh.

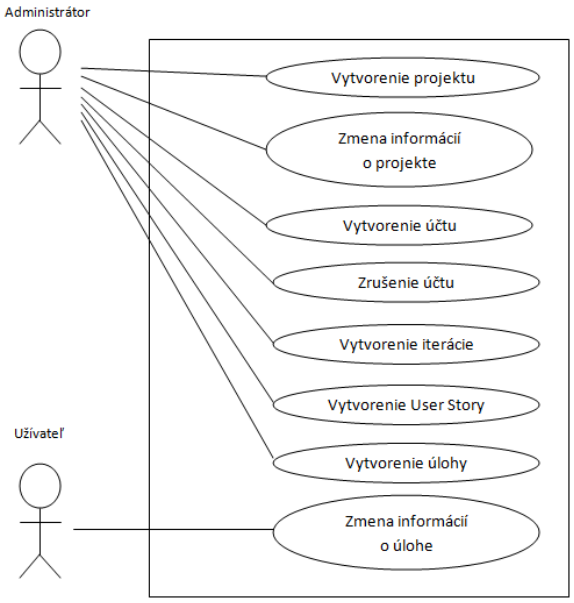
Skylink je najstaršou česko-slovenskou satelitnou platformou. Taktiež najrozšírenejším a najrýchlejšie sa rozrastajúcim poskytovateľom satelitného vysielania v Českej a Slovenskej republike. Poskytuje satelitné vysielanie pre viac ako 2,5 milióna domácností. [25]

4.2.4 Funkčná špecifikácia

Špecifikácia požiadaviek obsahuje Business Use Cases (BUC), ktoré abstraktne zachytávajú jednotlivé spúšťané scenáre. BUC sú písané všeobecne, z dôvodu zhrnutia funkcionality požadovanej všetkými spomínanými spoločnosťami. BUC môžu byť neskôr mapované na prípady užitia konkrétnych spoločností. Dôvod takejto úrovne abstrakcie je ten, že aplikácia je navrhovaná pre rôzne konfigurácie. Jednotlivé prípady užitia sú k dispozícii v prílohe diplomovej práce, od strany III.

Skupiny Business Use Cases:

- Správa projektov, obsahuje služby pre prácu s projektom. Aktérom je administrátor a skupina obsahuje služby:
  Vytvorenie projektu, Zmena informácií o projekte.
- Správa užívateľov, obsahuje služby pre správu užívateľov. Aktérom je administrátor a skupina obsahuje služby:
  Vytvorenie účtu, Zrušenie účtu
- Správa iterácií, obsahuje služby pre správu iterácií. Aktérom je administrátor a skupina obsahuje službu:
  Vytvorenie iterácie
- Správa obsahu iterácie, zahŕňa služby pre správu User Stories a taktiež služby pre správu úloh. Aktérmi sú administrátor alebo užívateľ a skupina obsahuje služby:
  Vytvorenie User Story, Vytvorenie úlohy, Zmena informácií o úlohe

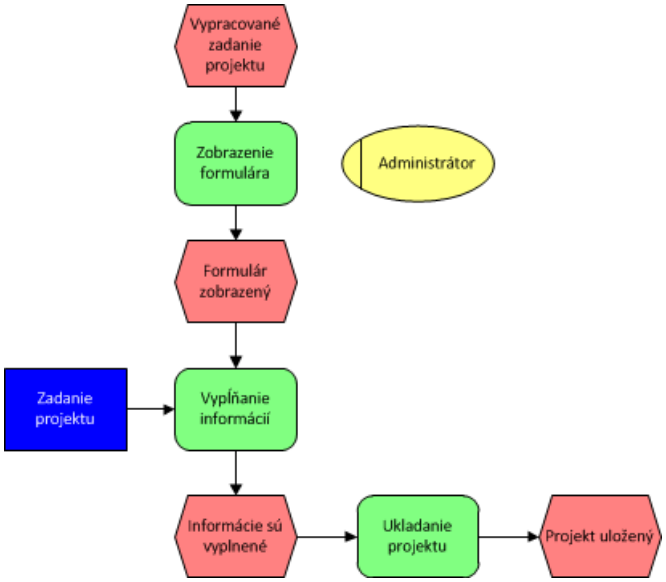


Obrázok 11: Business Use Case diagram: Práca s aplikáciou

4.2.5 Špecifikácia správania

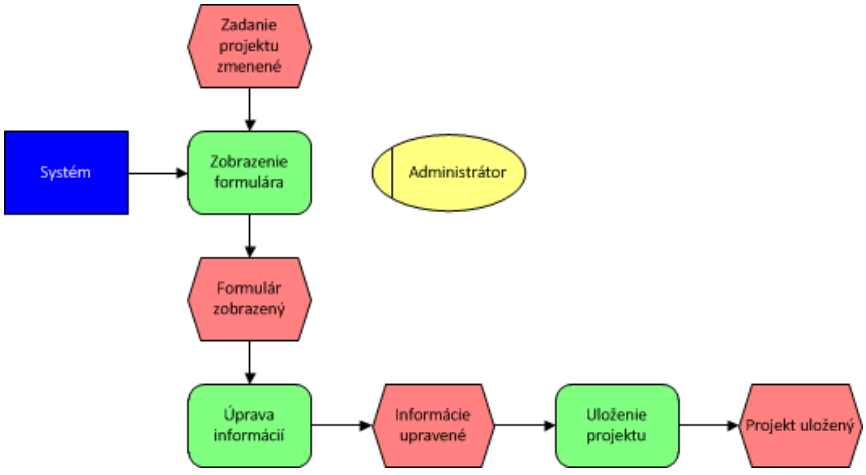
Pre špecifikáciu vybraných podprocesov som si vybral semiformálny prístup a to popis pomocou EPC (Event-driven Process Chain) diagramov doplnený o neformálny prístup textového popisu. Každý podproces je podrobnejšie popísaný pomocou biznis prípadu užitia ktorý je uvedený v prílohe práce.

Vytvorenie projektu začína udalosťou požiadanie na vytvorenie projektu v systéme. Systém zobrazí formulár, ktorý vyplní administrátor podľa zadania projektu. Po vyplnení informácií, administrátor zvolí možnosť uloženie projektu a ten je v systéme uložený. Prípad užitia je k dispozícii v prílohe D na strane III, pod označením BUC1.



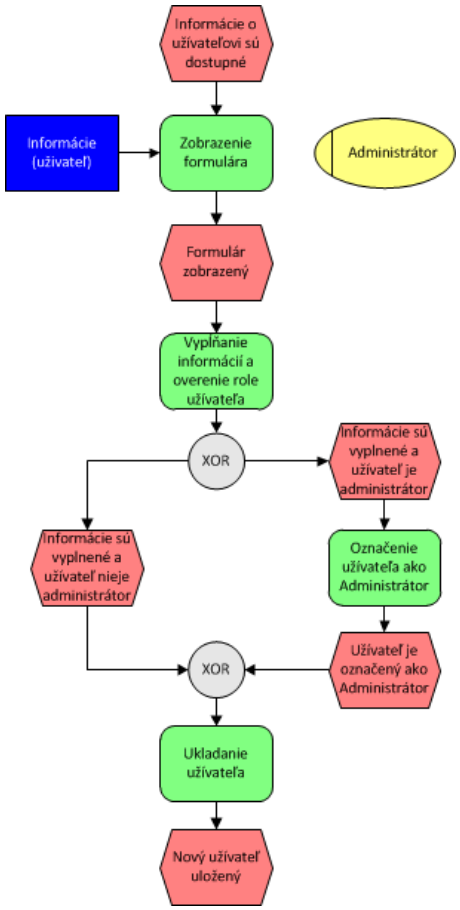
Obrázok 12: EPC diagram - vytvorenie projektu

Zmena informácií o projekte je vykonávaná administrátorom v prípade, že je o to požiadaný upraví potrebné informácie a zvolí možnosť uloženia projektu v systéme. Prípad užitia je k dispozícii v prílohe E na strane III, pod označením BUC2.



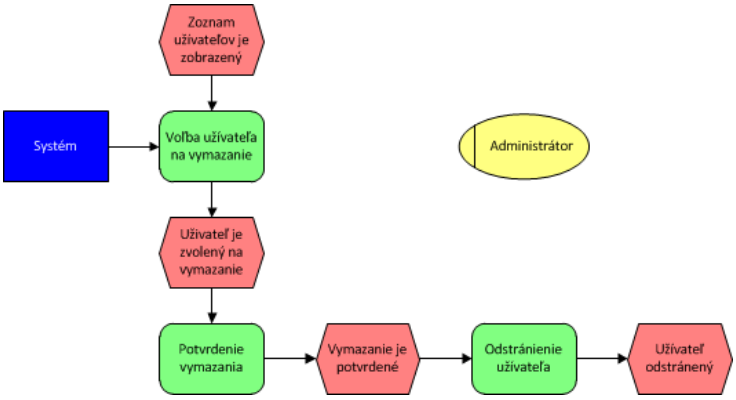
Obrázok 13: EPC diagram – zmena informácií o projekte

Vytvorenie účtu prebieha po požiadaní administrátora a poskytnutí mu dostatočných informácií o užívateľovi pre ktorého ma účet vytvoriť. Účet môže byť taktiež vytvorený ako administrátorsky. Prípad užitia sa nachádza v prílohe F na strane IV, pod označením BUC3.



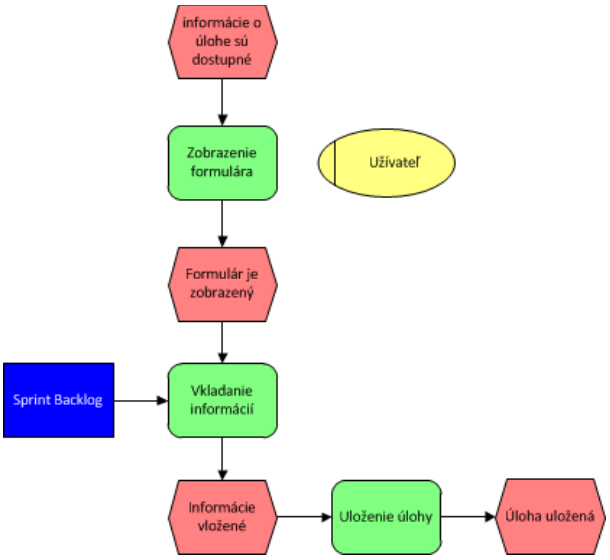
Obrázok 14: EPC diagram – vytvorenie účtu

Zrušenie účtu vykonáva administrátor a začína sa zobrazením zoznamu užívateľov a následne výberom daného užívateľa. Po potvrdení vymazania administrátorom, systém odstráni užívateľa z databázy. Prípad užitia sa nachádza v prílohe G na strane IV, pod označením BUC4.



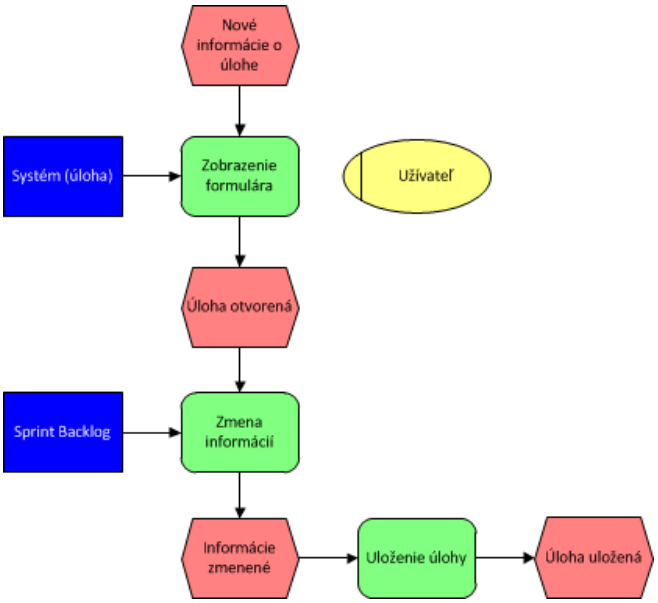
Obrázok 15: EPC diagram – zrušenie účtu

Vytvorenie úlohy vykonáva užívateľ a na začiatku je udalosť prijatá požiadavka pre vytvorenie novej úlohy. Následne sa zobrazí formulár v ktorom užívateľ vyplní informácie o úlohe a formulár odošle. Systém uloží novú úlohu do databázy. Vytváranie úloh prebieha väčšinou po naplánovaní iterácie, ale môžu byť vytvárané aj v priebehu iterácie. Podrobnejší priebeh zmeny informácií je popísaný pomocou prípadu užitia uvedeného v prílohe J na strane VI, pod označením BUC7.



Obrázok 16: EPC diagram – vytvorenie úlohy.

Zmena informácií o úlohe je vykonávaná užívateľom v prípade, že chce zmeniť niektoré hodnoty atribútov úlohy. Najčastejšia zmena sa predpokladá z dôvodu odpočítavania odpracovaných hodín na úlohe. Podrobnejší priebeh zmeny informácií je popísaný pomocou prípadu užitia uvedeného v prílohe K na strane VI, pod označením BUC8.



Obrázok 17: EPC diagram – zmena informácií o úlohe.

4.3 Softwarový proces ako biznis proces

Jedným zo špecifických cieľov tejto práce je navrhnúť softwarové procesy pre komerčné spoločnosti. Softwarové procesy sú založené na agilnej metodológií SCRUM. Pokiaľ hovoríme o Scrume ako frameworku, nejde o presne definovaný proces, ktorý by musel byť aj presne dodržaný. Je možné vybrať určité vhodné vlastnosti pre konkrétny proces a tieto vlastnosti následne integrovať. Navrhnuté procesy budú podporované implementovanou aplikáciou.

„Biznis proces je po častiach usporiadaná množina procedúr a aktivít, ktoré spoločne realizujú podnikateľský alebo strategický cieľ, obvykle v kontexte organizačnej štruktúry definujúcej funkcie rolí a ich vzťahov. Softwarový proces je taktiež určitý druh biznis procesu, avšak s určitými špecifickými vlastnosťami.“ [6]

4.4 Scrum proces

Všeobecne popísaný proces Scrum obsahuje základné črty, ktoré sú rovnaké pre jednotlivé špecifické procesy všetkých troch spoločností. Konkrétne procesy sa líšia v detailoch a tie sú bližšie popísané v podkapitolách s názvami spoločností a taktiež namodelované pomocou EPC diagramov, ktoré sú umiestnené v prílohe diplomovej práce.

Každá zo spoločností má vlastné požiadavky pre konfiguráciu implementovanej aplikácie, preto je vždy súčasťou návrhu procesov aj špecifická konfigurácia aplikácie, navrhnutá na základe požiadaviek danej spoločnosti.

4.4.1 Spracovanie požiadaviek

Táto fáza nasleduje po komunikácií so zákazníkom. Cieľom spracovania požiadaviek, je popísať získané požiadavky zrozumiteľnou formou, ako pre stranu zadávateľa tak pre stranu realizátora projektu. Spracovanie požiadaviek má na starosti Product Owner, ktorý komunikuje so zákazníkom a spracúva požiadavky do dokumentu nazvaného zadanie požiadaviek. Tento dokument je zároveň vstupným dokumentom pre fázu plánovania projektu.

Účastníci: Product Owner

Artefakt: Zadanie požiadaviek – obsahuje súbor informácií o požiadavkách, ktoré môžu byť reprezentované ako v textovej, tak v podobe grafických modelov.

4.4.2 Plánovanie projektu

Plánovanie projektu je fáza, ktorej účastníkom aj zákazník a ten má možnosť informovať o dôležitosti jednotlivých požiadaviek. Požiadavky sú spracované tak, aby ich bolo možné zaradiť do dokumentu nazývaného Product Backlog. Jednotlivé položky dokumentu sa nazývajú užívateľské príbehy (User Stories). Vypracovanie Product Backlogu má na starosti Product Owner. Po vypracovaní Product Backlogu musí Product Owner zoradiť jednotlivé položky podľa priority. Zoradenie podľa priority je

prípadne diskutované s členmi tímov, no konečné slovo pri tejto činnosti má Product Owner. Pokiaľ je Product Backlog odsúhlasený tímom, dokument je pripravený na použitie v ďalšej fáze vývoja.

Účastníci: Product Owner, Tím, Zákazník

Artefakty: Zadanie požiadaviek, Product Backlog

4.4.3 Plánovanie šprintu

Pre fázu plánovania šprintu je vstupným dokumentom schválený Product Backlog. Plánovanie prebieha na plánovacom mítingu a predchádza začiatku každého šprintu. Aj v tejto fáze je dôležitá účasť zákazníka, ktorý je podrobnejšie oboznámený s informáciami o vývoji v najbližšom šprinte a taktiež má možnosť odsúhlasiť položky, na ktorých sa bude pracovať. Členovia tímu majú za úlohu výber User Stories. Pre vybrané User Stories je tímom zvolená ich priorita. User Stories sú usporiadané podľa priority a vložené do dokumentu Sprint Backlog, čo je výstupným dokumentom fázy plánovania šprintu. Na plánovacom mítingu môžu byť rozložené jednotlivé User Stories na menšie úlohy, ktorým je nastavená aj priorita. Členovia tímu majú potom za úlohu rozdeliť si medzi seba User Stories, prípadne menšie úlohy. Scrum Master by mal dohliadať na to aby všetky úlohy boli primerane rozdelené medzi jednotlivých vývojárov.

Účastníci: Scrum Master, Tím, Zákazník

Artefakty: Product Backlog, Sprint Backlog

4.4.4 Priebeh šprintu

Vstupným dokumentom pre priebeh šprintu je Sprint Backlog. Šprint prebieha pokiaľ neuplynie čas vyhradený na jednu iteráciu. V priebehu iterácie prebieha vývoj a testovanie. Každých 24 hodín sa koná denný Scrum míting, na ktorom každý člen tímu odpovedá na tri otázky:

1. Čo si urobil od posledného mítingu?
2. Čo plánuješ robiť do ďalšieho mítingu?
3. Čo ti bráni v práci na zadaní?

Scrum míting je vedený Scrum Masterom, ktorý má za úlohu dohliadnuť na to aby každý člen odpovedal na spomínané otázky dostatočne presne a zrozumiteľne. Úlohou Scrum Mastera je taktiež nájsť riešenie prípadných problémov brániacich v ďalšom vývoji. V priebehu šprintu, nemôže dôjsť ku zmenám v Product Backlogu.

Účastníci: Scrum Master, Tím

Artefakty: Sprint Backlog

4.4.5 Ukončenie šprintu

Na konci každého šprintu sa koná stretnutie na ktorom prebieha zhodnotenie priebehu šprintu. Hodnotí sa aj to, koľko z naplánovaných úloh bolo dokončených. Úlohou stretnutia je aj zlepšovanie procesu. Hodnotiaci míting na konci šprintu je rozdelený na dve časti.

Sprint Review je prvá časť stretnutia na ktorom je Product Ownerovi a zákazníkovi prezentovaný výsledok prebehnutého šprintu. Dôležité je to, aby sa požiadavky zákazníka definované ako User Stories nerozchádzali z vytvoreným výsledným produktom.

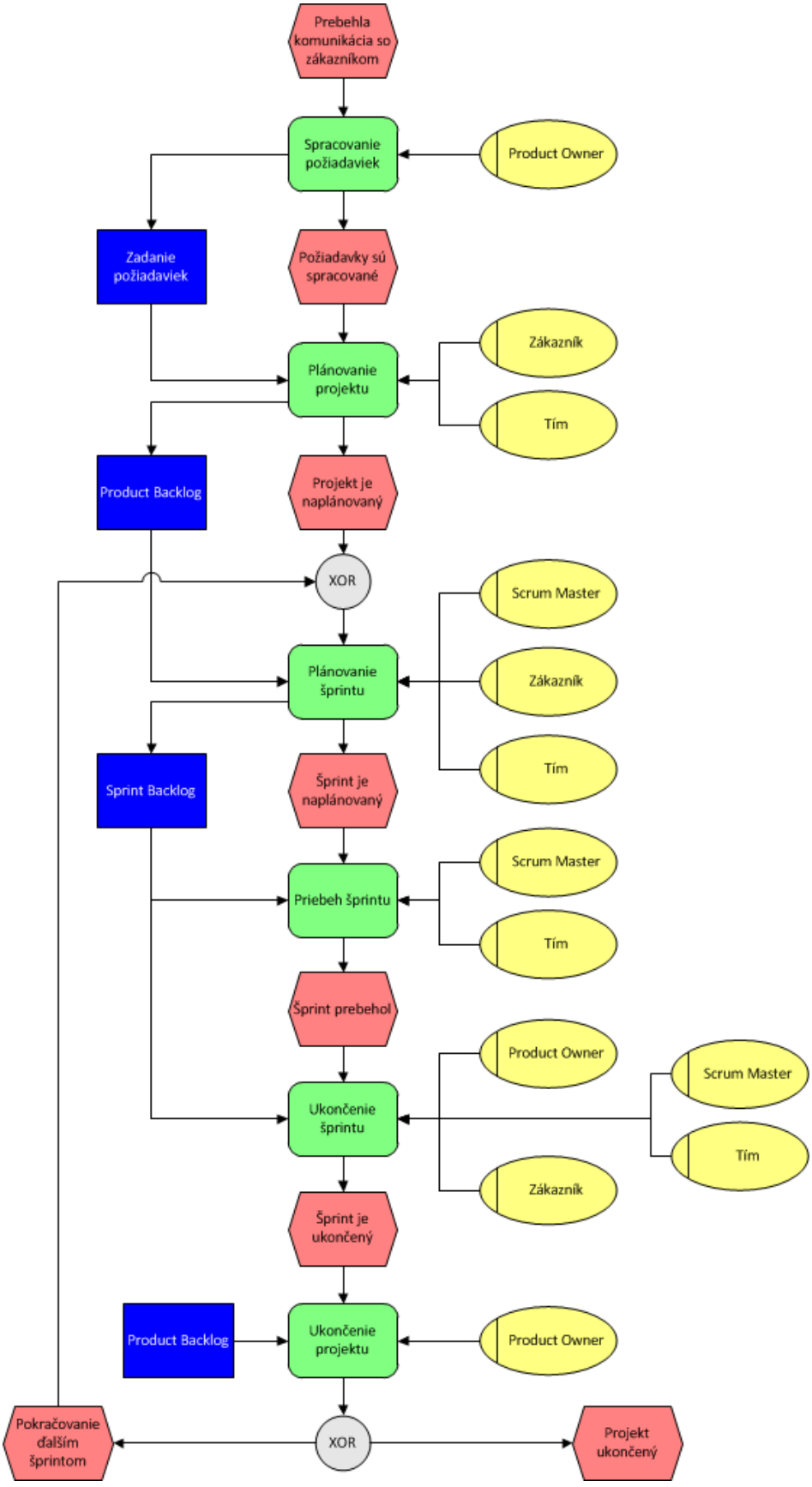
Sprint Retrospective je druhá časť stretnutia ktorej úlohou je nájsť nedostatky v prebehnutom šprinte a zlepšiť ďalší šprint, ktorý bude nasledovať.

Účastníci: Product Owner, Zákazník, Scrum Master, Tím

Artefakty: Sprint Backlog

4.4.6 Ukončenie projektu

Táto fáze Scrum procesu slúži na vyhodnotenie toho, či sú splnené všetky požiadavky Product Backlogu. V prípade, že sú splnené všetky požiadavky, projekt sa považuje za dokončený. V prípade že Product Backlog obsahuje ďalšie požiadavky, na ktorých splnenie sa ešte len čaká, bude pokračovať vo vývoji v ďalšom šprinte.



Obrázok 18: ECP diagram: Scrum proces

4.5 Navrhnutý proces pre spoločnosť STAPRO, s.r.o.

Softwarový proces pre spoločnosť STAPRO je špecifický v tom, že spoločnosť sa zaoberá vývojom kritických systémov pre zdravotníctvo, od ktorých fungovania závisí ľudské zdravie, alebo ľudské životy. Preto je nutné dbať na zvýšenú mieru testovania a dokumentovania navrhnutých a vyvíjaných riešení. Firma taktiež vzhľadom na typ vyvíjaných systémov neprijíma absolventov a menej skúsených vývojárov. Ďalším faktorom ovplyvňujúcim proces je možnosť vývojárov pracovať z domu 3 – 4 krát do mesiaca po 8 hodín. Spoločnosť pracuje na projektoch, ktoré sú vyvíjané aj viac ako 6 rokov a nie je možné stanoviť presnejšie plány na začiatku, preto proces nebude obsahovať plánovaciu iteráciu. Metrické jednotky sú ‚Človeko-hodiny', kde 1 človeko-hodina je čas trvania práce jedného človeka na zadanej úlohe.

4.5.1 Špecifikácia procesu

Plánovanie šprintu prebieha na stretnutí, ktoré trvá 6 hodín. Vedenie tohto stretnutia má za úlohu Scrum Master a cieľom je definovať čo všetko sa bude v priebehu nasledujúceho šprintu vyvíjať. Stretnutie je rozdelené na dve časti, kde každá časť trvá tri hodiny. Product Owner mal za úlohu pripraviť si dokument Product Backlog.

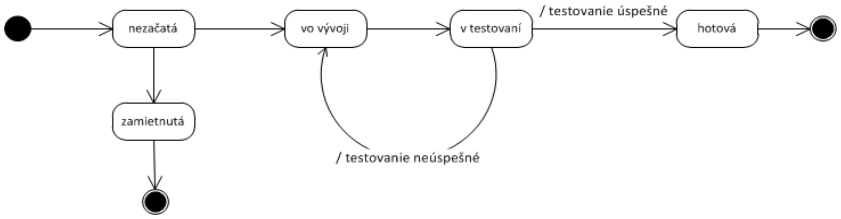
V prvej trojhodinovej fáze stretnutia Product Owner prezentuje Product Backlog pred vývojárskym tímom a zákazníkom, ktorý je pozvaný na plánovanie šprintu. V rovnakej fáze má tím za úlohu vybrať položky Product Backlogu, ktoré budú vyvíjané v nasledujúcom šprinte. Vybrané položky zároveň stanovujú ciele, ktoré má tím za úlohu dosiahnuť v priebehu nasledujúceho šprintu. Svoje výsledky bude tím prezentovať na konci šprintu, na stretnutí nazývanom Review meeting pred Product Ownerom a zákazníkom. Tím má možnosť navrhnúť zmeny v Product Backlogu, prípadne navrhnúť zloženie úloh na ktorých sa bude pracovať, no konečné rozhodnutie o Product Backlogu je na Product Ownerovi. Po analýze Product Backlogu nasleduje pol hodinová prestávka a po nej druhá fáza stretnutia.

Cieľom druhej fázy stretnutia je zostavenie dokumentu Sprint Backlog, ktorý obsahuje položky, na ktorých sa bude pracovať v priebehu šprintu. Product Owner musí byť dostupný aj počas druhej fázy, kde by mal byť schopný odpovedať na otázky tímu, ktoré sa týkajú Product Backlogu. Tím má momentálne sám rozhodovať o vývoji úloh vybratých predošlej fáze stretnutia. Na toto rozhodovanie nie je oprávnený nikto iný, ale iní účastníci stretnutia majú možnosť pozorovať priebeh alebo pokladať otázky členom tímu. Členovia tímu si rozdelia úlohy a odhadnú počet človeko – hodín pre každú úlohu. Výstupným dokumentom je Sprint Backlog, ktorý obsahuje zoznam úloh pre tím a odhad trvania práce na jednotlivých úlohách. Za administrátora aplikácie pre podporu SCRUM je zvolený Scrum Master a jeho úlohou v tejto fáze je podľa špecifikácie procesu plánovania, uvedeného v kapitole 4.1.1 vytvoriť nový projekt a vložiť doňho potrebné informácie.

Denný Scrum míting je stretnutie ktoré trvá 25 minút a zúčastňuje sa ho 8 členov vývojárskeho tímu a Scrum Master. Toto stretnutie prebieha v priestoroch spoločnosti a sú povinní zúčastniť sa všetci členovia tímu. Pokiaľ sa niektorý z členov tímu nemôže fyzicky zúčastniť

stretnutia, vzhľadom na to, že je umožnené niektoré dni pracovať mimo priestory firmy, jeho prítomnosť je zabezpečená pomocou video hovoru, za využitia aplikácie Skype. V prípade, že sa nemôže zúčastniť stretnutia vôbec, je povinný poveriť iného člena tímu, aby informoval ostatných o výsledkoch jeho práce a odpoveďami na otázky, ktoré sú kladené v priebehu Scrum mítingu. Účasť je požadovaná v presne stanovenom čase a meškanie na toto stretnutie je sankcionované pokutou pre člena tímu, ktorý mešká. Každý člen tímu odpovedá na otázky, ktoré sú uvedené vo všeobecnom Scrum procese a vyhradený čas na položenie všetkých troch otázok pre jedného člena a na ich odpovede sú 2 minúty. Scrum Master je povinný dohliadnuť na dodržanie času a relevantnosť odpovedí. Počas celého stretnutia je vždy umožnené v rovnakom čase rozprávať iba jednej osobe, ktorú ostatní počúvajú. Nie je umožnená konverzácia viacerých osôb naraz a o tom, kto bude rozprávať a odpovedať na otázky rozhoduje Scrum Master.

Priebeh šprintu je časovo ohraničená fáza, ktorá trvá 18 pracovných dní. V priebehu šprintu majú vývojári možnosť odrátavať nimi odpracovaný čas na jednotlivých úlohách a taktiež meniť stav úloh. Stav úlohy je určený podľa toho v ktorom stĺpci sa úloha nachádza. Každá vytvorená úloha je na začiatku v stave nezačatá. Úloha môže byť taktiež po vytvorení zamietnutá. Pokiaľ sa na úlohe začne pracovať zmení sa jej stav na ‚vo vývoji'. Po vývoji je úloha v testovaní. V prípade, že neprejde testovaním je vrátená do stavu vo vývoji. Pokiaľ testovaním prejde je presunutá do stavu hotová. Zmenu stavov, prípadne zmenu informácií o úlohe vykonáva užívateľ, ktorému je úloha priradená, prípadne je preňho viditeľná. Zmena informácií o úlohe prebieha podľa špecifikácie procesu implementácie, uvedenej v kapitole 4.1.2.



Obrázok 19: Stavový diagram - stavy úlohy

Ukončenie šprintu rozdelené do dvoch stretnutí z ktorých každé trvá tri hodiny. Prvé stretnutie sa nazýva Sprint Review Meeting a druhé Sprint Retrospecitve Meeting. Obe stretnutia prebiehajú jeden deň v priestoroch spoločnosti za účasti Product Ownera, Scrum Mastera, Tímu a zákazníka. Účasť všetkých spomenutých je povinná. V prípade že sa stretnutí nemôže fyzicky zúčastniť zákazník alebo člen tímu, je usporiadaná videokonferencia za pomoci aplikácie Skype.

Na stretnutí Sprint Review Meeting prezentujú členovia vývojárskeho tímu pred ostatnými zúčastnenými vyvinutú funkcionalitu. Položky Product Backlogu, ktoré neboli splnené nie sú prezentované.

4.5.2 Konfigurácia aplikácie

Kategórie úloh: Analýza, Konzultácie, Konzultácie v ambulancii, Vývoj, Prototyp, Testovanie

Stĺpce Scrum boardu: Nezačaté, Zamietnuté, Vo vývoji, V testovaní, Hotové

4.6 Navrhnutý proces pre spoločnosť SCOVECO, s.r.o.

Špecifiká procesu sú podmienené aj tým, že spoločnosť momentálne nedisponuje kancelárskymi priestormi a teda vývojári nepracujú na jednom mieste. Metrické jednotky sú ‚Človeko-hodiny', kde 1 človeko-hodina je čas trvania práce jedného človeka na zadanej úlohe.

4.6.1 Špecifikácia procesu

Plánovanie šprintu prebieha na stretnutí, ktoré trvá 4 hodiny. Vedenie tohto stretnutia má za úlohu Scrum Master a cieľom je definovať čo všetko sa bude v priebehu nasledujúceho šprintu vyvíjať. Stretnutie prebieha v zasadacej miestnosti prenajatej spoločnosťou a je rozdelené na dve časti, kde každá časť trvá dve hodiny. Product Owner má za úlohu pripraviť si dokument Product Backlog.

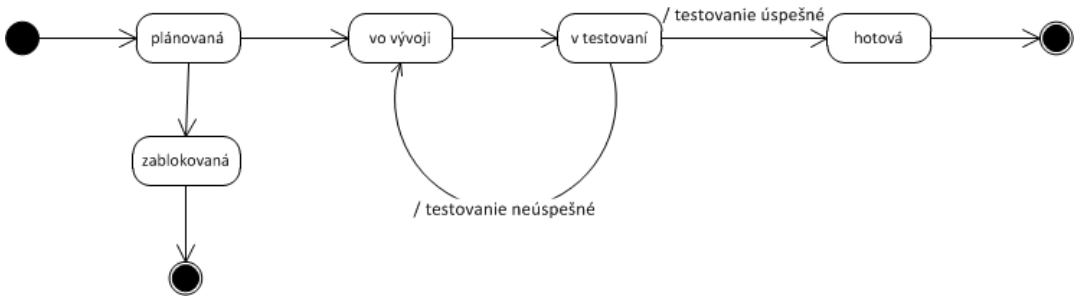
V prvej dvojhodinovej fáze stretnutia Product Owner prezentuje Product Backlog pred vývojárskym tímom a zákazníkom, ktorý je pozvaný na plánovanie šprintu. V rovnakej fáze má tím za úlohu vybrať položky Product Backlogu, ktoré budú vyvíjané v nasledujúcom šprinte. Vybrané položky zároveň stanovujú ciele, ktoré má tím za úlohu dosiahnuť v priebehu nasledujúceho šprintu. Svoje výsledky bude tím prezentovať na konci šprintu, na stretnutí nazývanom Review meeting pred Product Ownerom a zákazníkom. Tím má možnosť navrhnúť zmeny v Product Backlogu, prípadne navrhnúť zloženie úloh na ktorých sa bude pracovať, no konečné rozhodnutie o Product Backlogu je na Product Ownerovi. Po analýze Product Backlogu nasleduje štvrť hodinová prestávka a po nej druhá fáza stretnutia.

Cieľom druhej fázy stretnutia je zostavenie dokumentu Sprint Backlog, ktorý obsahuje položky, na ktorých sa bude pracovať v priebehu šprintu. Product Owner musí byť dostupný aj počas druhej fázy, kde by mal byť schopný odpovedať na otázky tímu, ktoré sa týkajú Product Backlogu. Tím má momentálne sám rozhodovať o vývoji úloh vybratých predošlej fáze stretnutia. Na toto rozhodovanie nie je oprávnený nikto iný, ale iní účastníci stretnutia majú možnosť pozorovať priebeh alebo klásť otázky členom tímu. Členovia tímu si rozdelia úlohy a odhadnú počet človeko – hodín pre každú úlohu. Výstupným dokumentom je Sprint Backlog, ktorý obsahuje zoznam úloh pre tím a odhad trvania práce na jednotlivých úlohách. Za administrátora aplikácie pre podporu SCRUM je zvolený Product Owner a jeho úlohou v tejto fáze je podľa špecifikácie procesu plánovania, uvedenej v kapitole 4.1.1, vytvoriť nový projekt a vložiť doňho potrebné informácie.

Denný Scrum míting je stretnutie, ktoré trvá 20 minút a zúčastňuje sa ho 6 členov vývojárskeho tímu a Scrum Master. Vzhľadom na to, že spoločnosť momentálne nedisponuje kancelárskymi priestormi, toto stretnutie prebieha za pomoci služby Google Hangout a sú povinní zúčastniť sa všetci členovia tímu. V prípade že sa nemôže niektorý z členov tímu zúčastniť stretnutia, je povinný poveriť iného člena tímu informovať ostatných o výsledkoch jeho práce a odpoveďami na otázky ktoré sú pokladané v priebehu Scrum mítingu. Účasť je požadovaná v presne stanovenom čase a meškanie na toto stretnutie je sankcionované pokutou pre člena tímu, ktorý mešká. Každý člen tímu odpovedá na otázky, ktoré sú uvedené vo všeobecnom Scrum procese a vyhradený čas na položenie všetkých troch otázok pre jedného člena a na ich odpovede sú 2 minúty. Scrum Master je povinný

dohliadnuť na dodržanie času a relevantnosť odpovedí. Počas celého stretnutia je vždy umožnené v rovnakom čase rozprávať iba jednej osobe, ktorú ostatní počúvajú. Nie je umožnená konverzácia viacerých osôb naraz a o tom kto bude rozprávať a odpovedať na otázky rozhoduje Scrum Master.

Priebeh šprintu je časovo ohraničená fáza, ktorá trvá 12 pracovných dní. V priebehu šprintu majú vývojári možnosť odrátavať svoj odpracovaný čas na jednotlivých úlohách a taktiež meniť stav úloh. Stav úlohy je určený podľa toho v ktorom stĺpci sa úloha nachádza. Každá vytvorená úloha je na začiatku v stave plánovaná. Úloha môže byť taktiež po vytvorení zamietnutá a vtedy je v stave zablokovaná. Pokiaľ sa na úlohe začne pracovať zmení sa jej stav na ‚vo vývoji'. Po dokončení sa úloha dostáva do stavu v testovaní. Pokiaľ testovanie prebehlo úspešne, je stav úlohy zmenený na hotová. V prípade neúspešného testovania je stav úlohy opäť zmenený na ‚vo vývoji'. Zmenu stavov, prípadne zmenu informácií o úlohe vykonáva užívateľ, ktorému je úloha priradená, prípadne je preňho viditeľná. Zmena informácií o úlohe prebieha podľa špecifikácie procesu implementácie, uvedenej v kapitole 4.1.2.



Obrázok 20: Stavový diagram – stavy úlohy

Ukončenie šprintu rozdelené do dvoch stretnutí z ktorých každé trvá 2 hodiny. Prvé stretnutie sa nazýva Sprint Review Meeting a druhé Sprint Retrospecitve Meeting. Obe stretnutia prebiehajú v jeden deň za účasti Product Ownera, Scrum Mastera, Tímu a zákazníka. Účasť všetkých spomenutých je povinná.

Na stretnutí Sprint Review Meeting prezentujú členovia vývojárskeho tímu pred ostatnými zúčastnenými vyvinutú funkcionalitu. Položky Product Backlogu, ktoré neboli splnené nie sú rezentované.

4.6.2 Konfigurácia aplikácie

Kategórie úloh: Analýza, Chyba, Vývoj, Testovanie

Stĺpce Scrum boardu: Plánované, Zablokované, Vo vývoji, V testovaní, Dokončené

4.7 Navrhnutý proces pre spoločnosť Skylink, s.r.o.

Software vyvíjaný spoločnosťou Skylink je kritický z hľadiska multimédií a biznisu. Vzhľadom na rozsiahlu užívateľskú základňu, je kladený dôraz na to, aby produkt, prípadne služba vo chvíli nasadenia neobsahovala chyby, alebo zásadné nedostatky. Metrické jednotky sú ‚Človeko-hodiny', kde 1 človeko-hodina je čas trvania práce jedného človeka na zadanej úlohe.

4.7.1 Špecifikácia procesu

Plánovanie šprintu prebieha na stretnutí, ktoré trvá 8 hodín. Vedenie tohto stretnutia má za úlohu Scrum Master a cieľom je definovať čo všetko sa bude v priebehu nasledujúceho šprintu vyvíjať. Stretnutie je rozdelené na dve časti, kde každá časť trvá štyri hodiny. Product Owner má za úlohu pripraviť si dokument Product Backlog.

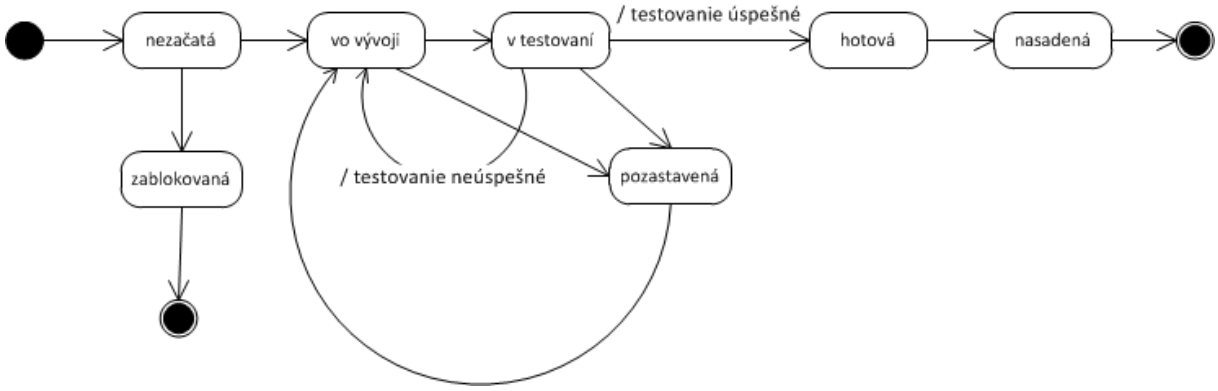
V prvej štvorhodinovej fáze stretnutia Product Owner prezentuje Product Backlog pred vývojárskym tímom a zákazníkom, ktorý je pozvaný na plánovanie šprintu. V rovnakej fáze má tím za úlohu vybrať položky Product Backlogu, ktoré budú vyvíjané v nasledujúcom šprinte. Vybrané položky zároveň stanovujú ciele, ktoré má tím za úlohu dosiahnuť v priebehu nasledujúceho šprintu. Svoje výsledky bude tím prezentovať na konci šprintu, na stretnutí nazývanom Review meeting pred Product Ownerom a zákazníkom. Tím má možnosť navrhnúť zmeny v Product Backlogu, prípadne navrhnúť zloženie úloh na ktorých sa bude pracovať. Konečné rozhodnutie o Product Backlogu je však na Product Ownerovi. Po analýze Product Backlogu nasleduje pol hodinová prestávka a po nej druhá fáza stretnutia.

Cieľom druhej fázy stretnutia je zostavenie dokumentu Sprint Backlog, ktorý obsahuje položky, na ktorých sa bude pracovať v priebehu šprintu. Product Owner musí byť dostupný aj počas druhej fázy, kde by mal byť schopný odpovedať na otázky tímu, ktoré sa týkajú Product Backlogu. Tím má momentálne sám rozhodovať o vývoji úloh vybratých v predošlej fáze stretnutia. Na toto rozhodovanie nie je oprávnený nikto iný, ale iní účastníci stretnutia majú možnosť pozorovať priebeh, alebo klásť otázky členom tímu. Členovia tímu si rozdelia úlohy a odhadnú počet človeko – hodín pre každú úlohu. Výstupným dokumentom je Sprint Backlog, ktorý obsahuje zoznam úloh pre tím a odhad trvania práce na jednotlivých úlohách.

Denný Scrum míting je stretnutie ktoré trvá 30 minút a zúčastňuje sa ho 9 členov vývojárskeho tímu a Scrum Master. Toto stretnutie prebieha v priestoroch spoločnosti a sú povinní zúčastniť sa všetci členovia tímu. V prípade, že sa člen tímu nemôže zúčastniť stretnutia, je povinný poveriť iného člena tímu objasnením ostatných s výsledkami jeho práce a odpoveďami na otázky, ktoré sú kladené v priebehu Scrum mítingu. Účasť je požadovaná v presne stanovenom čase a meškanie na toto stretnutie je sankcionované pokutou pre člena tímu ktorý mešká. Každý člen tímu odpovedá na otázky, ktoré sú uvedené vo všeobecnom Scrum procese a vyhradený čas na položenie všetkých troch otázok pre jedného člena a na ich odpovede sú 2 minúty. Scrum Master je povinný dohliadnuť na dodržanie času a relevantnosť odpovedí. Počas celého stretnutia je vždy umožnené

v rovnakom čase rozprávať iba jednej osobe, ktorú ostatní počúvajú. Nie je umožnená konverzácia viacerých osôb naraz a o tom kto bude rozprávať a odpovedať na otázky rozhoduje Scrum Master.

Priebeh šprintu je časovo ohraničená fáza, ktorá trvá 14 pracovných dní. V priebehu šprintu majú vývojári možnosť odrátavať svoj odpracovaný čas na jednotlivých úlohách a taktiež meniť stav úloh. Stav úlohy je určený podľa toho v ktorom stĺpci sa úloha nachádza. Každá vytvorená úloha je na začiatku v stave nezačatá. Pokiaľ sa na úlohe začne pracovať zmení sa jej stav na ‚vo vývoji'. V prípade že na začiatku marketing rozhodne o zablokovaní úlohy, na úlohe sa nikdy nezačne pracovať. Po vývoji je úloha v testovaní. V prípade že neprejde testovaním je vrátená do stavu vo vývoji. Pokiaľ testovaním prejde je presunutá do stavu hotová a následne do stavu nasadená. Úloha v stavoch vo vývoji alebo v testovaní môže byť kedykoľvek pozastavená. V prípade pokračovania práce na pozastavenej úlohe je opäť jej stav zmenený na ‚vo vývoji'. Zmenu stavov, prípadne zmenu informácií o úlohe vykonáva človek, ktorý je poverený správou úloh v aplikácii. Zmena informácií o úlohe prebieha podľa špecifikácie procesu implementácie, uvedenej v kapitole 4.1.2.



Obrázok 21: Stavový diagram – stavy úlohy

Ukončenie šprintu rozdelené do dvoch stretnutí z ktorých každé trvá štyri hodiny. Prvé stretnutie sa nazýva Sprint Review Meeting a druhé Sprint Retrospecitve Meeting. Obe stretnutia prebiehajú jeden deň v priestoroch spoločnosti za účasti Product Ownera, Scrum Mastera, Tímu a zákazníka. Účasť všetkých spomenutých v priestoroch spoločnosti je povinná.

Na stretnutí Sprint Review Meeting prezentujú členovia vývojárskeho tímu pred ostatnými zúčastnenými vyvinutú funkcionalitu. Položky Product Backlogu, ktoré neboli splnené nie sú rezentované.

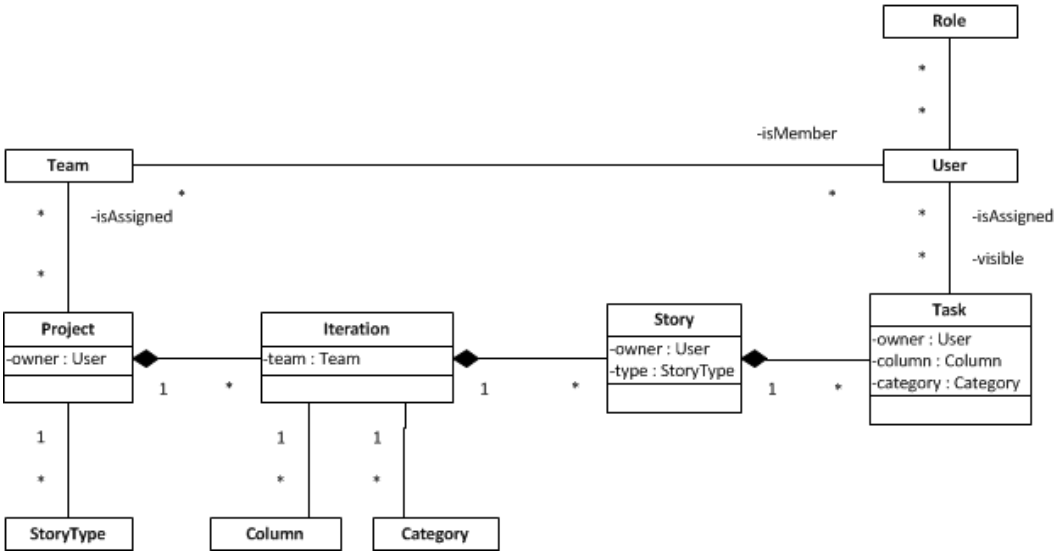
4.7.2 Špecifická konfigurácia aplikácie

Kategórie úloh: Analýza, Konzultácie, Vývoj, Incident, Servisná požiadavka, Testovanie

Stĺpce Scrum boardu: Nezačaté, Zablokované, Vo vývoji, V testovaní, Dokončené, Pozastavené

4.8 Analýza aplikácie

Model analýzy slúži na správne pochopenie štruktúry systému. Na obrázku nižšie, sú uvedené základné analytické triedy aplikácie spolu s kľúčovými atribútmi. Vzťahy medzi triedami sú zachytené pomocou triedneho diagramu:



Obrázok 22: Triedny diagram analytických tried aplikácie

Project: Táto trieda reprezentuje softwarový projekt. Každý projekt má svojho vlastníka (ownera), predpokladá sa že vlastníkom bude Product Owner, ktorý je reprezentovaný triedou User. Každý projekt obsahuje atribúty, názov, popis, dátum začiatku, dátum predpokladaného ukončenia, názov organizácie, a stav v ktorom sa práve nachádza (otvorený alebo uzavretý). Projektu môže byť pridelených niekoľko vývojárskych tímov. Na triedu Project je naviazaná trieda StoryType, ktorá reprezentuje typ užívateľského príbehu. Jeden projekt môže obsahovať niekoľko typov príbehov.

Iteration: SCRUM je založený na iteratívnom prístupe. Iterácia v analyzovanej aplikácií reprezentuje šprint. Existencia iterácie je závislá na existencii projektu, preto vzťah medzi týmito dvoma triedami je spojenie pomocou agregácie, kde jeden projekt môže obsahovať niekoľko iterácií. Iterácia obsahuje atribúty, názov, dátum začiatku, dátum ukončenia a stav. Povinným atribútom iterácie je tím, ktorý je reprezentovaný triedou Team. Na triedu Iteration sú naviazané triedy Column, ktorá reprezentuje stĺpce a Category reprezentuje kategórie úloh. Jedna iterácia môže obsahovať niekoľko stĺpcov a taktiež niekoľko kategórií.

Story: Reprezentuje užívateľský príbeh a je závislá od existencie iterácie, preto vzťah medzi iteráciou a užívateľským príbehom je zachytený pomocou agregácie. Jedna iterácia môže obsahovať niekoľko príbehov. Povinnými atribútmi príbehu sú vlastník, reprezentovaný triedou User a typ, reprezentovaný triedou Type. Príbeh obsahuje názov, popis a kritéria splnenia požadovanej úlohy.

Task: Úloha je podrobné zadanie pre konkrétneho vývojára. Existencia úlohy je závislá na existencií užívateľského príbehu a ako to bolo v predošlých vzťahoch medzi triedami, aj tu je vzťah zachytený pomocou agregácie, kde jeden príbeh môže obsahovať niekoľko úloh. Povinným atribútom

úlohy je vlastník, ktorý je reprezentovaný triedou User. Ďalšími povinnými atribútmi sú kategória úlohy, typu Category a stĺpec typu Column. Ostatné atribúty úlohy sú popis, číslo, odhadovaný čas, zostávajúci čas a priorita. Úlohe môže byť priradených niekoľko užívateľov, pre ktorých je viditeľná.

Trieda User reprezentuje užívateľov uložených v systéme. Každému užívateľovi môže byť priradených niekoľko rolí, reprezentovaných triedou Role. Užívateľ môže byť členom niekoľkých tímov, reprezentovaných triedou Team.

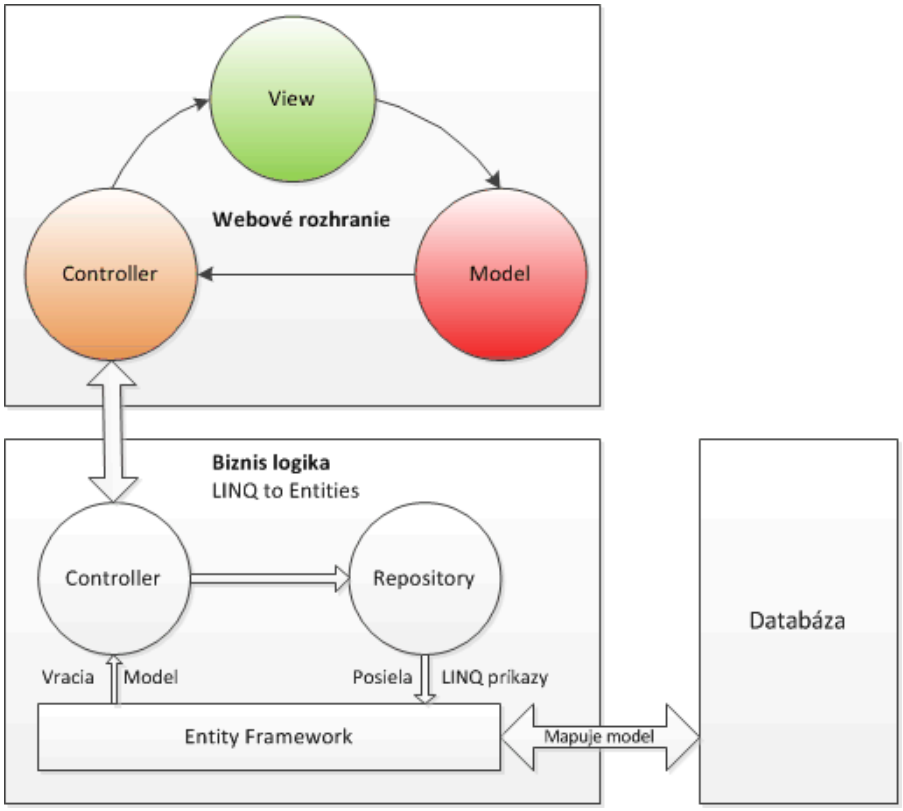
Úplný databázový model aplikácie sa nachádza v prílohe diplomovej práce.

4.9 Návrh aplikácie

Cieľom návrhu aplikácie bolo nájsť webové riešenie, ktorého užívateľské rozhranie bude vhodne oddelené od aplikačnej logiky a to z dôvodu jednoduchšej zmeny za iné rozhranie ktoré môže byť v budúcnosti k dispozícii. Ďalším cieľom bolo vytvoriť databázovo nezávislé riešenie, ktoré bude možné pripojiť na rôzne typy známych databáz.

4.9.1 Architektúra

Aplikácia je vytvorená v rámci troch väčších celkov, z ktorých jedným je webové rozhranie aplikácie, ďalej biznis logika a databáza. Aplikácia je tvorená v rámci dvoch oddelených projektov v nástroji Microsoft Visual Studio, z ktorých jeden je projekt Webová aplikácia - MVC a druhým projektom je knižnica s biznis logikou. Jednotlivé vrstvy a komunikácia medzi nimi sú znázornené na obrázku 23.



Obrázok 23: Ukážka navrhnutých vrstiev a komunikácie medzi nimi.

Webové rozhranie je vytvorené pomocou trojvrstvovej architektúry MVC (Model – View - Controller). Pre tvorbu užívateľského rozhrania, optimalizovaného pre mobilné zariadenia iPad a Samsung Galaxy bola zvolená technológia ASP.NET spolu s knižnicou jQuery Mobile. Užívateľské rozhranie sa pomocou vrstvy Controller pripája ku biznis logike aplikácie.

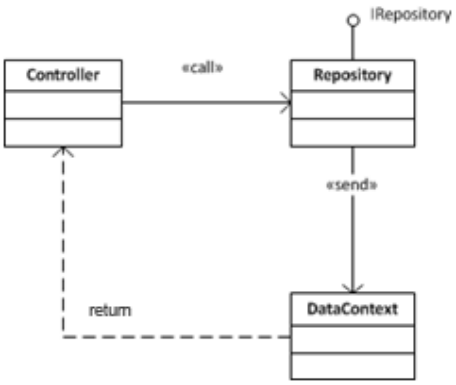
Biznis logika je tvorená aplikačnou vrstvou, ktorá komunikuje s vrstvou Entity Framework. Aplikačná logika komunikuje s Entity Frameworkom, ktorý sa stará o mapovanie databázy a pomocou ORM LINQ. Aplikačná logika je tvorená triedou Controller, ktorá volá metódy triedy Repository a ten následne posiela potrebné LINQ príkazy na dátovú vrstvu. Entity Framework, dátovej vrstvy komunikuje s SQL databázou a vracia vrstve Controller požadované dáta. Takáto kombinácia zvolených technológií zabezpečuje nezávislosť aplikácie na databázovej platforme. Spojenie aplikačnej logiky a dátovej vrstvy je vytvorené podľa návrhového vzoru Repository, ktorý je v kontexte použitia s LINQ to Entities podrobnejšie rozobratý v nasledujúcej kapitole.

Voľba ľubovoľnej databázovej platformy je možná za predpokladu nahratia potrebných ovládačov pre komunikáciu Entity Frameworku s databázou. Momentálne aplikácia funguje v spojení s databázou MySQL.

4.9.2 **Návrhový vzor Repository**

Pre komunikáciu aplikačnej logiky s dátovou vrstvou bol zvolený návrhový vzor Repository a to z dôvodu, že umožňuje oddelenie aplikačnej logiky od dátovej vrstvy a tak napríklad aj lepšiu možnosť testovania logickej časti aplikácie. Návrhový vzor som implementoval spolu s technológiou LINQ to Entities. V rámci časti biznis logiky aplikácie je implementovaná vrstva Controller, ktorá volá metódy triedy Repository. [22]

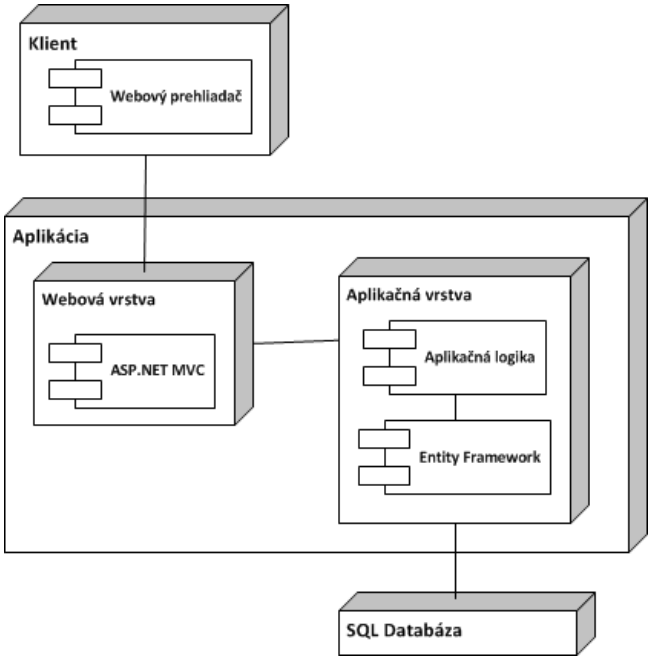
Controller volá metódu, ktorou si pýta od Repository objekt alebo kolekciu objektov a to bez nutnosti poznať typ databázy, alebo spôsob pripojenia ku nej. Repository sa správa ako keby pracoval s dátovými kolekciami, pričom je možné jednoduché pridávanie, úprava a mazanie objektov. Repository následne posiela LINQ príkazy pre Entity Framework Data Context, ktorý vracia Controlleru požadovaný objekt alebo kolekciu objektov. Na obrázku nižšie sú znázornené jednotlivé triedy a vzťahy medzi nimi.



Obrázok 24: Ukážka implementácie návrhového vzoru Repository v kontexte Linq to Entities.

4.9.3 **Model nasadenia**

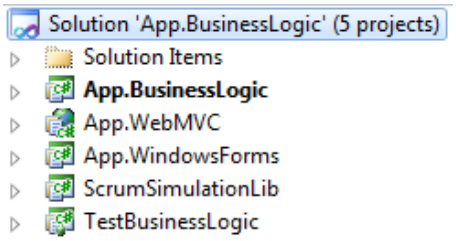
Na obrázku 25 je znázornený diagram nasadenia, ktorý zobrazuje štruktúru prostriedkov. Klient pristupuje cez internet, pomocou webového prehliadača ku komponente ASP.NET webovej vrstvy aplikácie, ktorá je umiestnená na webovom serveri. V rámci aplikácie komunikuje webová vrstva s aplikačnou vrstvou. Aplikačná vrstva ďalej komunikuje s SQL databázou. Aplikačná vrstva obsahuje dve komponenty, aplikačnú logiku a Entity Framework. Aplikačná logika komunikuje s Entity Frameworkom, ktorý mapuje SQL databázu.



Obrázok 25: Diagram nasadenia

5 Implementácia

Implementácia aplikácie je rozdelená na dve hlavné časti. Prvú časť tvorí implementácia biznis logiky, ktorá bola implementovaná ako knižnica v C#.NET. Druhú časť tvorí užívateľské rozhranie. V prvej fáze bolo z testovacích dôvodov vytvorené zjednodušené užívateľské rozhranie vo Windows Forms. V druhej fáze, po otestovaní kľúčových funkcií biznis logiky bolo vytvorené webové užívateľské rozhranie v ASP.NET.



Obrázok 26: Ukážka štruktúry projektov v nástroji Microsoft Visual Studio 2010.

5.1 Zvolené nástroje a technológie pre vývoj aplikácie

Na implementáciu aplikácie bol využitý nástroj spoločnosti Microsoft, Visual Studio 2010 Professional. Pre prácu s MS SQL databázou bol použitý taktiež nástroj od spoločnosti Microsoft, SQL Server Management Studio a ďalej nástroj pre prácu s MySQL databázou MySQL Workbench.

Pre tvorbu užívateľského rozhrania bolo zvolené prostredie ASP.NET 4.0 spolu s frameworkom MVC 2, ktorý zabezpečuje dodržanie trojvrstvovej architektúry webovej časti aplikácie. Front - end časť bola vytvorená aj s pomocou javascriptovej knižnice jQuery mobile 1.1.0, optimalizovanej pre najpoužívanejšie mobilné zariadenia.

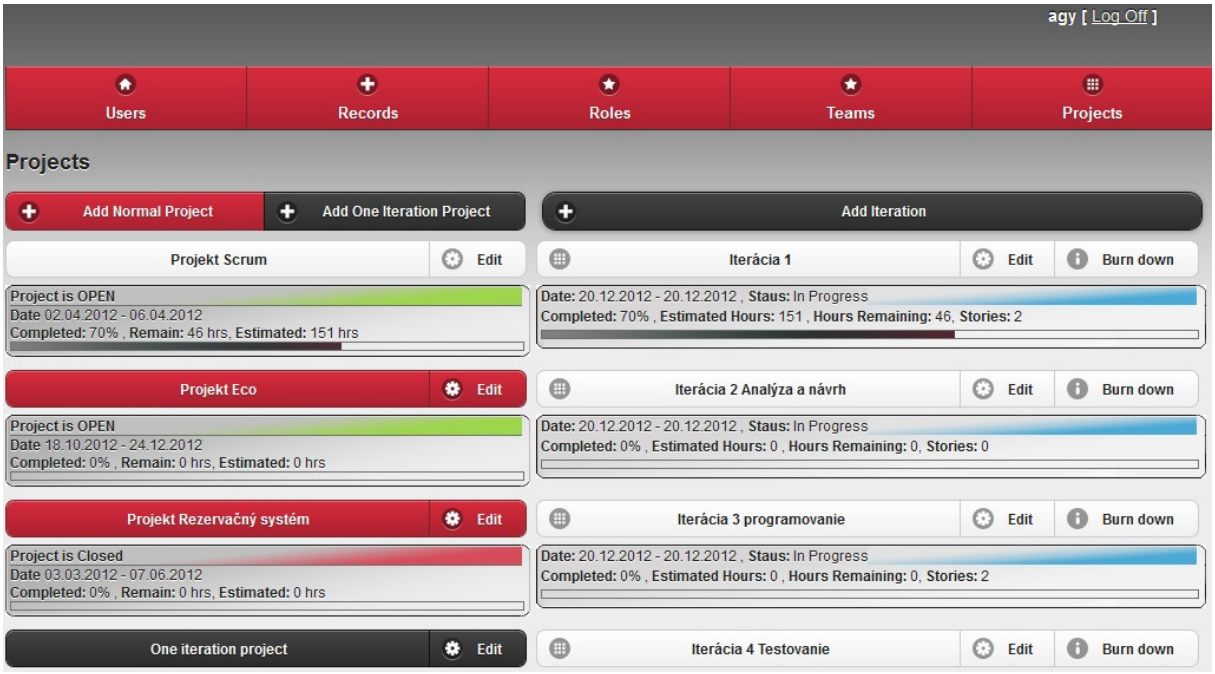
Biznis logiku aplikácie je vytvorená, ako samostatný projekt vo Visual Studiu 2010, ktorý je typu Class Library v jazyku C# s frameworkom .NET 4.0. Ďalšou súčasťou biznis logiky je LINQ, ktorý slúži na dotazovanie Entity frameworku. Entity Framework sa stará o mapovanie MySQL databázy ku ktorej je aplikácia momentálne pripojená s pomocou ovládača .Net Framework Data Provider for MySQL. Voľbou vhodného ovládača a zmenou nastavenia pripojenia je možné použitie aj iných typov databáz.

Systémové požiadavky pre beh aplikácie sú:

- Diskový priestor 0,5 GB, RAM 100 MB, MySQL / MS SQL / Oracle SQL (30MB)
- Operačný systém Windows Web Server 2008
- Podporované technológie:
  - ASP.NET 4.0
  - ASP.NET Web Services
  - ASP.NET MVC 2
  - ASP.NET AJAX
  - LINQ to ADO.NET

5.2 Časti aplikácie

Prvá časť, ktorá sa zobrazí po načítaní webovej aplikácie je prihlasovanie. Po prihlásení je užívateľ presmerovaný na zoznam projektov (v sekcii Projects), ktoré sú zobrazené v ľavej časti stránky. Po otvorení niektorého z projektov sa v pravej časti zobrazí zoznam iterácií prislúchajúcich otvorenému projektu. Pokiaľ má užívateľ systémové práva administrátora, sú preňho viditeľné položky užívatelia, záznamy, role, tímy a projekty. Podrobnejšie sú tieto časti popísané v podkapitole 5.2.2 Administrácia. Na obrázku 27 je ukážka časti zoznamu projektov a iterácií, ktorá sa zobrazí užívateľovi po prihlásení.



Obrázok 27: Ukážka sekcie Projects – zoznam projektov a iterácií.

5.2.1 Užívateľské rozhranie

Užívateľské rozhranie je primárne optimalizované pre tablety iPad a Samsung Galaxy s rozlíšením obrazovky, šírky 1024 pixelov. Z dôvodu takejto optimalizácie bola zvolená JavasScript knižnicu jQuery mobile verzie 1.1.0. Užívateľské rozhranie je možné plnohodnotne zobraziť aj na klasickom monitore s rozlíšením o šírke väčšej ako 1024 pixelov.

Dôležitou časťou v rámci tvorby užívateľského rozhrania bolo vytvorenie ‚Scrum boardu', čo je tabuľa využívaná v Scrume na zobrazenie prehľadu úloh, ich stavu, prípadne typu. Úlohy sú pripevnené na klasickej tabuli v podobe farebných papierových lístkov, kde farba zobrazuje typ danej úlohy (analýza, návrh, vývoj) a umiestnené v stĺpcoch, ktoré reprezentujú ich stav (nezačaté, vo vývoji, hotové). Primárnym cieľom tabule je prehľadné zobrazenie. V aplikácii obsahuje takúto tabuľu každá iterácia. Kvôli zachovaniu základných princípov Scrumu je možná voľba rôznych stĺpcov a rôznych typov úloh pre každú iteráciu samostatne.

Úlohy na ‚Scrum boarde' aplikácie je možné presúvať dotykom na display a pri klasickom počítači pohybom myši. Pre zabezpečenie možnosti presúvania úloh som použil knižnicu v javascripte

s názvom jQuery mobile Drag and Drop. Túto knižnicu som následne upravil do aktuálnej podoby, kde dochádza k ukladaniu aktuálnej pozície presunutej úlohy do databázy.

Scrum board obsahuje zoznam užívateľských príbehov označených ako Stories, ktoré je možné ľubovoľne pridávať v rámci iterácie. Úlohy každého príbehu sú umiestnené v záložke, ktorá sa zobrazí po kliknutí na príbeh. Užívateľský príbeh obsahuje informácie o svojom type, veľkosti, poznámkach a taktiež obsahuje stĺpce, ktoré boli definované pri vytváraní iterácie.

Obrázok 28: Ukážka Scrum boardu aplikácie.

Po presunutí úlohy do iného stĺpca volá funkcia Display, knižnice Drag and Drop, metódu ChangePosition webovej služby aplikácie s názvom BoardService. Táto metóda ma za úlohu uloženie aktuálnej pozície úlohy. Metódy spolu komunikujú pomocou formátu JSON, kde JavaScript posiela parametre, identifikátor úlohy a identifikátor aktuálneho stĺpca.

```
function Display(id, tempCurrentDrag) {
        var pageUrl = '../../WebServices/BoardService.asmx'
        $.ajax({
        type: "POST",
        url: pageUrl + "/ChangePosition",
        data: "{'position': '" + id + "', 'idTask': '"
                + tempCurrentDrag + "'}",
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: OnSuccessCall,
        error: OnErrorCall
        });
```

Výpis 1: Ukážka zdrojového kódu funkcie Display z knižnice jQuery mobile Drag and Drop.

Burndown Chart je ďalším zaujímavým prvkom užívateľského rozhrania. Čiary grafu zobrazujú ideálny (modrá čiara) a aktuálny (červená čiara) priebeh vývoja, viď. Obrázok 29. Graf umožňuje prehľadné zobrazenie priebehu prác na iterácii. Pre grafické vykresľovanie grafu som použil komponentu Chart, primárne integrovanú vo Visual Studiu 2010. Graf je možné zobraziť individuálne, pre každú iteráciu. Súradnice bodov grafu sú generované s pomocou knižnice v C#.NET pre simuláciu softwarového procesu, zloženom na báze Scrumu.

Obrázok 29: Ukážka Burndown grafu.

5.2.2 Administrácia

Prístup do administračnej časti má užívateľ so systémovým oprávnením administrátora. Toto oprávnenie môže nastavovať administrátor pri vytváraní nového užívateľa, prípadne zmenou nastavení existujúceho užívateľa. Administrátor má k dispozícii prehľad užívateľov, log - záznamov, užívateľských rolí, vývojárskych tímov a projektov.

Users je sekcia, ktorá obsahuje zoznam všetkých užívateľov uložených v systéme. Je možné vytvoriť nový užívateľský účet, alebo upraviť či vymazať existujúci účet. Pri vytváraní alebo úprave účtu je možné priradiť užívateľovi rolu, alebo vývojársky tím. Pri úprave účtu je taktiež možné vynulovať odpracované hodiny daného užívateľa. Ukážka v prílohe L na strane VII.

Records je sekcia log – záznamov o úpravách vytvorených úloh. V prípade že užívateľ zmení počet odpracovaných hodín na danej úlohe, systém vytvorí záznam z informáciami práci na úlohe. Administrátor má možnosť odstrániť záznamy z tejto sekcie. Ukážka v prílohe M na strane VII.

Roles je sekcia ktorá obsahuje prehľad užívateľských rolí. Tieto role môžu byť priradené viacerým užívateľom. Role obsahujú názov. Je možné ich vytvárať, upraviť alebo odstrániť. Ukážka v prílohe N na strane VIII.

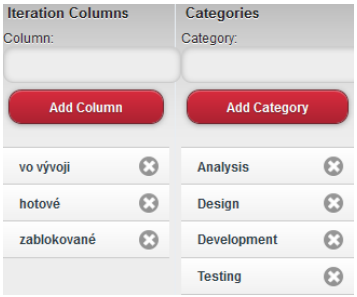
Teams je sekcia obsahujúca prehľad tímov. Každý tím má svoj názov a môže byť priradený viacerým užívateľom podobne ako role. V tejto sekcii je možné vidieť ktorí užívatelia sa nachádzajú v daných tímoch. Tím obsahuje názov a meno spoločnosti. Tímy je taktiež možné pridávať, upraviť, alebo odstrániť. Ukážka v prílohe O na strane VIII.

5.3 Problémy pri implementácii a ich riešenie

Pri tvorbe webového užívateľského bolo nutné vyriešiť niekoľko problémov. V tejto podkapitole sú predstavené dva zaujímavé problémy a taktiež ich riešenia.

5.3.1 Dynamicky generovaný zoznam

Zoznam na obrázku 30 je generovaný dynamicky pomocou javascriptu. Pridávané položky cez textové pole sú automaticky pridávané do dátových štruktúr a pomocou javascriptu zobrazené do html kódu stránky.



Obrázok 30: Ukážka záznamu generovaného javascriptom.

Problém nastáva vtedy, keď potrebujeme uložiť hodnoty zoznamu (napr. vo vývoji, hotové a zablokované) ako samostatné položky do databázy.

```
<ul>
        <li> vo vývoji </li>
        <li> hotové </li>
        <li> zablokované </li>
</ul>
```

Výpis 2: Ukážka html kódu generovaného javascriptom.

Riešenie spočíta v ukladaní týchto hodnôt do skrytého textového poľa, oddelené čiarkou a zároveň ako viditeľný html výstup na stránke. Hlavným cieľom je zabezpečiť pomocou javascriptu, to aby boli v skrytom textovom poli vždy rovnaké hodnoty, ktoré sú viditeľné aj na stránke. Po odoslaní formulára je textová hodnota zo skrytého poľa spracovaná a rozdelená na samostatné položky ktorú sú uložené do databázy.

5.3.2 Checkbox

Ďalší problém, ktorý sa vyskytol, spočíva v potrebe ukladať vlastnosti reprezentované na HTML stránke v podobe checkboxov. Po odoslaní bolo potrebné rozlíšiť, ktoré prvky (checkboxy) sú označené ako ‚checked' a tie, ktoré nie sú takto označené.

Riešenie problému spočíva v načítaní kolekcie objektov s rovnakým názvom prvkov, napríklad „teamObjects". Potom je táto skupina načítaná a spracovaná po jednotlivých položkách.

```
string[] teamObject = collection["teamObjects"].Split(',');
```

Výpis 3: Ukážka zdrojového kódu v C#, ktorý má za úlohu spracovanie kolekcie prvkov.

6 Simulácia softwarového procesu

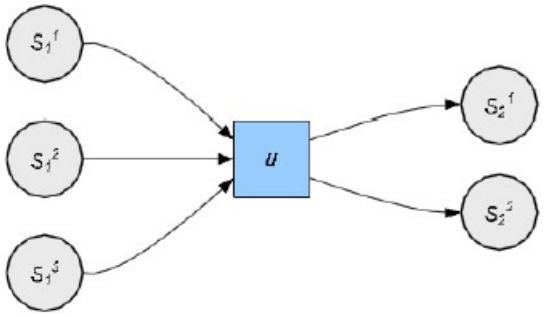
Táto kapitola sa venuje predstaveniu simulačného prístupu Discrete-Event Simulation a taktiež popisu metódy a knižnice pre simuláciu softwarového procesu, ktorá bola vytvorená ako doplnková časť diplomovej práce a je založená práve na DES prístupe.

6.1 Formálne metódy

Formálne metódy popisujú a modelujú systémy v podobe matematických modelov. Veľkou výhodou matematických modelov je schopnosť verifikácie a simulácie správania systému. Existuje niekoľko formálnych metód ktoré je možné použiť, ako napríklad konečné stavové automaty, Petriho siete alebo Workflow siete. Pre formalizáciu boli vybraté práve Petriho siete. [6]

6.1.1 Petriho siete

Petriho siete sú grafický a matematický modelovací nástroj, aplikovateľný na široké spektrum systémov. Z historického hľadiska, koncept Petriho sietí má pôvod v dizertácii človeka s menom Carl Adam Petri, z roku 1962. Petriho sieť je určitý druh orientovaného grafu, spolu s počiatočným značením M_0. Základný graf N, Petriho siete je orientovaný, vážený, bipartitný graf zahŕňajúci dva typy uzlov, nazývané miesta a prechody. V grafickej reprezentácií, miesta sú nakreslené ako kruhy, prechody ako obdĺžniky, alebo štvorce. [6]



Obrázok 31: Ukážka Petriho siete. [6]

V modelovaní, pri používaní konceptu podmienok a udalostí, miesta reprezentujú podmienky a prechody reprezentujú udalosti. Prechod (udalosť), má presne určitý počet vstupných a výstupných miest. Prítomnosť tokenu[4] v určitom mieste je interpretovaná, ako splnenie pravdivosti podmienky spojenej s miestom. V inej interpretácii, *k* tokenov je vložených do určitého miesta a indikujúcich to, že je dostupných *k* dátových položiek alebo zdrojov. [6] Niektoré typicky používané reprezentácie prechodov a ich vstupných a výstupných miest, sú zobrazené v tabuľke 2.

[4] Token (anglický výraz) vyjadruje značku v mieste Petriho siete, graficky reprezentovanú bodkou.

Tabuľka 2: Niektoré typické interpretácie prechodov a miest.

Vstupné miesta	Prechod	Výstupné miesta
Vstupná podmienka	Udalosť	Výstupná podmienka
Vstupné dáta	Výpočtový krok	Výstupné dáta
Vstupné signály	Signálový procesor	Výstupné signály
Potrebné zdroje	Úloha	Uvoľnené zdroje

Stav alebo značenie Petriho siete, sú zmenené podľa nasledujúcich pravidiel:

- Prechod t bude aktívny, pokiaľ každé miesto p, prechodu t je označené najmenej s w(p,t) počtom tokenov, kde w(p,t) je váha hrany z miesta p do prechodu t.
- Aktivovaný prechod môže, alebo nemusí byť vykonaný.
- Vykonanie aktívneho prechodu t odstráni w(p,t) tokenov z každého vstupného miesta p, prechodu t a pridá w(p,t) tokenov každému výstupnému miestu p prechodu t, kde w(p,t) je váha hrany z t do p.

Formálna definícia hovorí, že Petriho sieť je 5-tica, $PN = (P,T,F,W,M_0)$, kde:

- $P = \{p_1, p_2, ... p_n\}$ je konečná množina miest
- $T = \{t_1, t_2, ... t_n\}$ je konečná množina prechodov
- $F \subseteq (P \times T) \cup (T \times P)$ je množina hrán
- $W: F \rightarrow \{1,2, 3, ... \}$ je funkcia váženia
- $M_0: P \rightarrow \{0, 1, 2, 3, ... \}$ je počiatočné značenie
- $P \cap T = \emptyset$ a $P \cup T = \emptyset$

Štruktúra Petriho siete $N = (P,T,F,W)$, bez počiatočného značenia je označená písmenom N. Petriho sieť s počiatočným značením je označená ako (N,M_0). [6]

6.2 Disciplína modelovania a vytvorenie modelu

Na začiatku, ešte pred využitím formálnych modelov softwarových procesov pre simuláciu a optimalizáciu je potrebné definovať disciplínu modelovania. Modelovaný systém často reprezentuje realitu, alebo iný umelý, komplexný systém a model je zjednodušenie - abstrakcia. Tri funkcie abstrakcie pre modelovanie: [13]

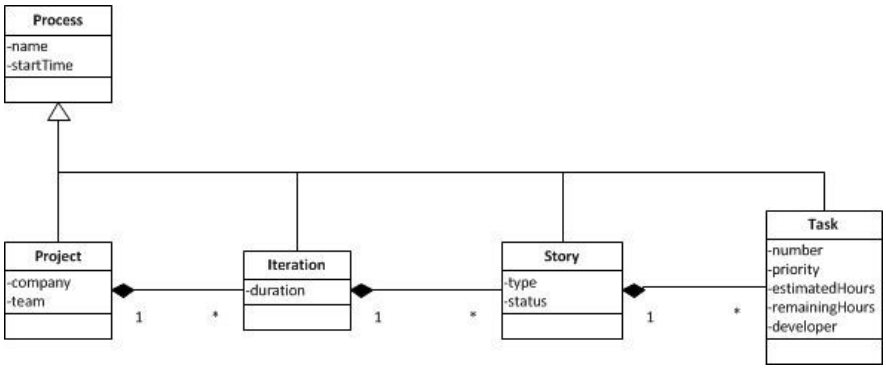
- Agregácia: entita skladajúca sa z niektorých ďalších entít, je modelovaná ako jedna agregovaná entita.
- Klasifikácia: je identifikovaná trieda entít zdieľajúcich niektoré spoločné vlastnosti.
- Generalizácia: nová trieda entít zdieľajúcich spoločné vlastnosti je definovaná abstrakciou z tých vlastností v ktorých sú odlišné.

Máme definované 3 základné typy entít, ktoré sú dôležité pre SCRUM proces: projekt, iterácia a úloha. Projekt zahŕňa kolekciu iterácií. Iterácia je v tomto prípade časovo ohraničená. SCRUM je

založený na iteratívnom prístupe, ktorý využíva časovo ohraničené cykly – šprinty. V tomto prípade sa za šprint považuje práve iterácia. V procese vývoja zohrávajú hlavné role vývojári. V modely je každý vývojár charakterizovaný sadov atribútov, ako napríklad skúsenosti alebo úväzok vo firme.

6.2.1 Entitný model

- Projekt: Táto entita reprezentuje softwarový projekt. Ten je charakterizovaný parametrami: Name, StartTime, Company, Team. Vývoj projektu prebieha cez určitý počet iterácií. Projekt preto zahŕňa kolekciu iterácií.
- Iterácia: Iterácia obsahuje parametre, ako svoj vlastný identifikátor (Name), SartTime, Duration a taktiež zahŕňa kolekciu užívateľských príbehov (Stories).
- Story: Reprezentuje požiadavku pre vývoj systému. Obsahuje parametre Name, Type, Status.
- Úloha: Každá úloha má svoj názov, prioritu, odhadovaný a zostávajúci čas a taktiež vývojára.



Obrázok 32: Časť triedneho diagramu, entít Scrumu.

6.3 Simulácia

Simulácia softwarového procesu sa stáva čoraz populárnejšou v komunite softwarových inžinierov. Môže poskytnúť veľmi užitočné informácie o slabých miestach v agilnom softwarovom procese. Vývoj simulačného modelu je relatívne nie drahý spôsob získavania informácií v porovnaní s experimentovaním pri reálnom vývoji softwarových projektov.

Model je abstrakcia reálneho alebo konceptuálneho komplexného systému. Model je navrhnutý pre zobrazenie významných vlastností, charakterizujúcich systém, ktorý by mal byť ďalej skúmaný, predpovedaný alebo riadený. Tento model zahŕňa niektoré, ale nie všetky aspekty modelovaného systému.

Simulačný model je počítačový model, ktorý spracúva charakteristický popísaný dynamický systém. Jednou z hlavných motivácií pre vývoj simulačného modelu alebo použitia inej modelovacej metódy je to, že sa jedná o nie drahý spôsob ako nadobudnúť dôležité informácie o systéme, kde je dôležitá cena, riziko, prípadne logistika. [28]

6.3.1 Discrete-Event Simulation

DES umožňuje modelovanie systému a jeho vývoja v čase. Silnou stránkou tohto prístupu, je schopnosť modelovať náhodné prekážky (udalosti) a predpovedať vplyv komplexnej interakcie medzi týmito udalosťami.

Entity sú elementy z reálneho sveta a môžu byť buď to dočasné alebo stále. Logické vzťahy spájajú odlišné entity a práve oni sú kľúčovou časťou simulačného modelu. Ďalšou kľúčovou časťou simulácie je riadenie priebehu simulácie. Riadenie priebehu je zodpovedné za riadenie času. Časovač je použitý na dodržiavanie časových úsekov. Riadenie priebehu bude riadiť logické vzťahy medzi entitami a taktiež čas. Riadenie priebehu simulácie je základ, pre poskytnutie dynamického správania sa modelu založenom na čase. [13]

DES je efektívny prístup vtedy, pokiaľ je proces zobrazený ako sekvencia aktivít, ako pri výrobnej linke, kde sa položky alebo entity presúvajú zo stanice na stanicu. Diskrétny model jednoducho reprezentuje rad a pokiaľ nie sú dostupné zdroje, môže dôjsť aj k meškaniu. Navyše každá entita má svoje vlastnosti. [27]

6.3.2 Simulácia agilného procesu

Simulácia prebieha za použitia vstupných parametrov uvedených v tabuľke 3. Tieto hodnoty sú založené na reálnych dátach z vývoja ‚klinických' modulov informačného systému pre zdravotníctvo, ktorý je implementovaný v univerzitnej nemocnici a väčších klinikách.

Konštantné hodnoty sú počet úloh a trvanie iterácie. Funkcia pre generovanie času trvania jednotlivých úloh zahŕňa dva vstupné parametre: Minimum Estimated Hours a Maximum Estimated Hours. V závislosti na vstupných parametroch, ktoré sú uvedené v tabuľke 3, je automatický generovaná kolekcia úloh.

Tabuľka 3: Vstupné parametre simulácie

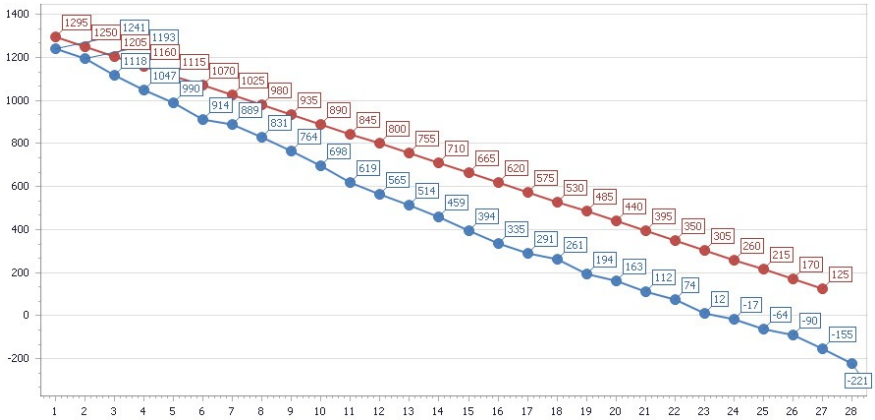
Parameter	Hodnota	Ostatné
Počet úloh	100	
Počet vývojárov	5	Tabuľka 4.
Minimálne trvanie úlohy	4 hodiny	
Maximálne trvanie úlohy	14 hodín	
Trvanie jednej iterácie	10 dní	

Jeden alebo viacero aktérov vykonávajú jednotlivé aktivity procesu. Tím, ktorého aktivity sú simulované je zložený z vývojárov a títo sú aktérmi v rámci simulácie (Tabuľka 3.). Títo aktéri sú charakterizovaní tromi hlavnými atribútmi: prvá vlastnosť skúsenosť, znamená prax v rokoch, druhá vlastnosť dostupnosť, znamená pracovný úväzok v rámci jedného mesiaca.

Tabuľka 4: Zoznam vývojárov

Id	Skúsenosť	Dostupnosť
1	3	140
2	4	150
3	4	250
4	5	180
5	6	180

Burndown Chart na obrázku 33 zobrazuje ideálny (lineárny) a simulovaný priebeh procesu. Graf obsahuje dve čiary. Prvá (červená) lineárne klesajúca čiara zobrazuje predpovedaný priebeh procesu za optimálnych podmienok a v ideálnom prostredí spoločnosti, ktorá vyvíja software na báze frameworku SCRUM. Druhá (modrá) čiara zobrazuje simulovaný priebeh procesu, zahŕňajúci skúsenosti a dostupnosť vývojárov. Dôležitým faktorom vplývajúcim na priebeh procesu je generovanie prekážok. Tieto prekážky sú prebraté z reálneho priebehu softwarových procesov. Sú to napríklad výpadok prúdu alebo internetu. Prekážky sú generované v periódach s exponenciálnym náhodným rozdelením. Trvanie týchto prekážok je nastavené podľa normálového rozdelenia.



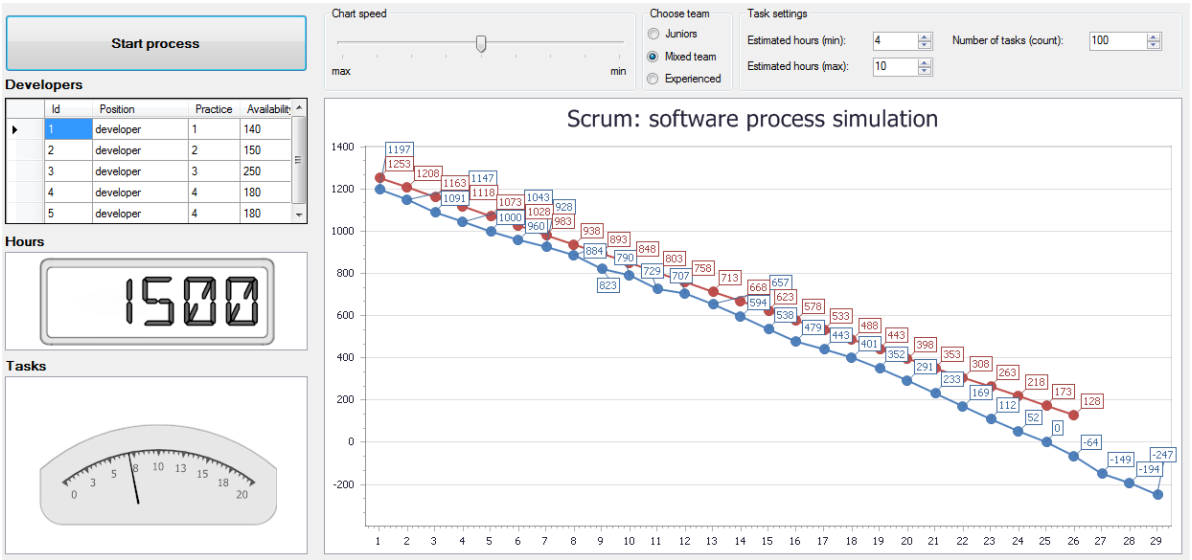
Obrázok 33: Ukážka grafu Burndown Chart.

Ako ukazuje obrázok 33, celkový odhadovaný čas práce na projekte je okolo 1295 človeko - hodín. Ideálny odhadovaný priebeh procesu trvá okolo 27 dní. Odhadovaný priebeh procesu bol prekročený v simulovanom priebehu o 221 človeko - hodín. Simulovaný proces trval 23,5 dní.

6.4 Záver simulácie

V tejto kapitole je predstavený prístup pre modelovanie a simuláciu softwarového procesu, založeného na agilnej metóde SCRUM a formálnych modelovacích nástrojoch. Po krátkom úvode sú popísané základy formálneho modelovania, disciplíny modelovania a simulačná metóda.

Pre otestovanie simulačnej metódy bol vyvinutý softwarový nástroj na platforme C# .NET. Nástroj simuluje priebeh softwarového procesu založený na Scrume. Výstupom simulácie je Burndown Chart a počet človeko – hodín trvania simulácie. Ukážka aplikácie, ktorá bola vyvinutá s užívateľským rozhraním Windows Forms je zobrazená na obrázku 34.



Obrázok 34: Ukážka nástroja pre simuláciu Scrum procesu.

Záver

Hlavnou myšlienkou tejto práce bola implementácia aplikácie softwarovej podpory SCRUM. Návrh agilných softwarových procesov pre komerčné softwarové spoločnosti a zaškolenie personálu týchto spoločností, pre používanie aplikácie v spojení s dodržiavaním navrhnutými procesmi.

V úvode tejto práce, boli bližšie popísané najrozšírenejšie agilné prístupy, ktoré sú momentálne používané v komerčnej sfére vývoja softwaru. Následne boli porovnané s tradičnými metódami vývoja a podrobnejšie rozobraté vlastnosti a výhody agilného frameworku SCRUM. Na základe nadobudnutých znalostí o agilných metodológiách a na základe konzultácií so zástupcami troch komerčných spoločností, bol vytvorený zoznam požiadaviek. Boli navrhnuté softwarové procesy a taktiež navrhnutá aplikácia pre podporu softwarových procesov. Nakoniec bola navrhnutá aplikácia implementovaná a následne boli zaškolení na jej používanie kľúčoví zástupcovia dvoch softwarových spoločností, viď. príloha Q na strane X a príloha R na strane XI. Zadanie bolo splnené v plnom rozsahu a navyše bol dodaný modul pres simuláciu softwarového procesu.

6.5 Dosiahnuté výsledky a zhodnotenie

Ako už bolo spomenuté cieľom práce bola implementácia aplikácie. Vývoj aplikácie bol smerovaný tak, aby bolo možné využívať nástroj plnohodnotne aj v mobilných zariadeniach, ale aj ako databázovo nezávislé riešenie.

- Navrhnutý nástroj umožňuje plánovanie vývoja softwarových projektov, evidenciu vývojárov, vývojárskych tímov a prideľovanie jednotlivých úloh vývojárom.
- Podľa požiadaviek sme pre prácu s databázou zvolili ORM framework LINQ v kombinácií s technológiou ADO.NET. Ku testovaniu komunikácie s rôznymi typmi databázových platforiem boli využité databázy MySQL, MS SQL a ORACLE.
- Vzhľadom na to, že pri vývoji bola využitá knižnica jQuery mobile a grafické rozhranie je optimalizované pre mobilné zariadenia. Primárna optimalizácia umožňuje plnohodnotné zobrazenie na mobilných zariadeniach iPad a Samsung Galaxy.
- Aplikácia bola rozšírená o modul pre simuláciu softwarového procesu.

Implementovaná aplikácia bola predstavená kľúčovým zástupcom dvoch softwarových spoločností. Po predstavení aplikácie došlo ku zaškoleniu týchto zástupcov. Školenie bolo zamerané na oblasť metodiky vývoja softwaru SCRUM, spoločne s metodikami o správnom dodržiavaní softwarového procesu, ktorý bol pre konkrétne spoločnosti navrhnutý.

V rámci zberu požiadaviek bolo dôležité ujasniť si so zadávateľmi aktuálny priebeh procesov v jednotlivých firmách. Na základe zozbieraných informácií bolo možné navrhnúť nové procesy a taktiež vypracovať analýzu a návrh aplikácie. Prácu na vývoji aplikácie môžem zhodnotiť ako prínosnú a zaujímavú. Zoznámil som sa s novými metodológiami a technológiami. Mal som možnosť nahliadnuť do fungovania komerčných softwarových spoločností a taktiež pre ne navrhnúť agilné

softwarové procesy. Implementovaná aplikácia bude v budúcnosti využívaná nielen na plánovanie softwarových projektov, ale vďaka rozširujúcemu modulu aj na simulácie softwarových procesov.

6.6 Plány do budúcnosti

Vo vývoji aplikácie sa plánuje pokračovať, vzhľadom na to že zadávatelia prichádzajú s novými podnetmi a nápadmi na rozšírenia. V rámci doktorského štúdia je plánované pokračovať vo vývoji a vylepšovaní rozširujúceho modulu pre simulácie softwarových procesov, ako aj vo výskume v oblastí simulácií agilných procesov. Niektoré konkrétnejšie plány sú stručne spomenuté v nasledujúcich bodoch:

- Integrácia do Visual Studia, Eclipse a taktiež do verzovacích systémov podľa požiadaviek tímov (Team Foundation Server, SVN, CVS).
- Modelovanie závislostí medzi úlohami.
- Rekonfigurácia procesov.
- Reportovaci nástroj – designer reportov.
- Meranie a evidenciu času, ktorý majú jednotlivý vývojári k dispozícii na prezentáciu svojich výsledkov v rámci denných mítingov.
- Vývoj metód pre simulácie softwarových procesov založených na agilných metodológiách a tak umožniť efektívnejšie a presnejšie plánovanie vývoja.
- Personalizácia v simuláciách – rastúci presun parametrov a presun skúseností.

6.7 Licencia

Zdrojový kód aplikácie softwarovej podpory SCRUM je vlastníctvom autora, Radoslava Štrbu a Vysokej školy Báňskej – Technickej univerzity Ostrava. Zdrojový kód nie je súčasťou prílohy diplomovej práce a akékoľvek ďalšie použitie pre akademické účely je možné iba zo súhlasom autora, Radoslava Štrbu. Akékoľvek použitie pre komerčné účely je zakázané v plnom rozsahu, je však možné písomne požiadať o uvoľnenie tejto licencie pre komerčné účely Vysokú školu Báňskú - Technickú univerzitu Ostrava a autora Radoslava Štrbu. Zdrojové kódy ďalších modulov, ktoré sú voľne k dispozícii sú súčasťou prílohy a sú voľne použiteľné v plnom rozsahu podľa licencie týchto diel.

Použitá literatúra

[1] HAAPASALO, T. Using Open-Source Solutions in Agile Software Development, 2007. Master's Thesis at HELSINKI UNIVERSITY OF TECHNOLOGY

[2] Agile Modeling [online]. 2013 [cit. 2013-04-02]. Available from WWW: <www.agilemodeling.com >.

[3] Agile Alliance [online]. 2013 [cit. 2013-04-02]. Available from WWW: <www.agilealliance.com >.

[4] PUURUNEN, V. Assessing Agility of Software Development Teams, 2010. Master's Thesis at UNIVERSITY OF OULU

[5] K. SCHWABER, Agile Project Management with Scrum: Microsoft Press, 2004. ISBN 073-56-1993-x

[6] VONDRÁK, I. Methods of Business Modeling. VŠB-TUO, Czech Republic, Ostrava, 2004.

[7] VONDRÁK, I., KOŽUSZNIK, J., OCHODKOVÁ, E.: Methods for software systems specification. VŠB-TUO, Czech Republic, Ostrava, 2006

[8] VONDRÁK, I. Software Engineering. VŠB-TUO, Czech Republic, Ostrava, 2002.

[9] AWAD, M.: A Comparison between Agile and Traditional Software Development Methodologies, 2005. Master's Thesis at THE UVIVERSITY OF WESTERN AUSTRALIA

[10] MACDONALD, M., FREEMAN, A., SZPUSZTA, M. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně, kniha 1. Brno: Zoner Press, 2011. ISBN 978-80-7413-131-8

[11] MACDONALD, M., FREEMAN, A., SZPUSZTA, M. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně, kniha 2. Brno: Zoner Press, 2011. ISBN 978-80-7413-145-5

[12] RESIG, John. jQuery: Kuchařka programátora. Brno: Computer Press, 2010. ISBN 978-80-251-3152-7

[13] ŠTRBA, R., KOŠINÁR, M., ČERNOHORSKÝ, J.: Knowledge Modeling and Simulations of Agile Software Processes based on Machine Learning Methods. Appear in the Proceedings of the 26th European Japanese Conference on Information Modelling and Knowledge Bases (EJC 2013), Nara, Japan 2013.

[14] KOŠINÁR, M.: Design and Utilization of Knowledge Bases for Software Processes, 2010. Diplomová práce na FEI VŠB TU Ostrava

[15] ARLOW, J., NEUSTADT, I. UML2 a unifikovaný proces vývoje aplikací. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9

[16] COCKBURN, Alistair. Use Cases: Jak efektivně modelovat aplikace. Bron: Computer Press, 2005. ISBN 80-251-0721-3

[17] SeeNowDo [online]. 2013 [cit. 2013-04-02]. Dostupné z WWW: <www.seenowdo.com>.

[18] PangoScrum [online]. 2013 [cit. 2013-04-02]. Dostupné z WWW: <www.pangoscrum.com>.

[19] ScrumDo [online]. 2013 [cit. 2013-04-02]. Dostupné z WWW: <www.scrumdo.com>.

[20] Scrum Master [online]. 2013 [cit. 2013-04-02].
Dostupné z WWW: <www. scrummaster.com.au>.

[21] Version One [online]. 2013 [cit. 2013-04-02].
Dostupné z WWW: < www.versionone.com>.

[22] MSDN – The Repository Pattern [online]. 2013 [cit. 2013-09-04].

Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/ff649690.aspx>.

[23] STAPRO [online]. 2013 [cit. 2013-04-02].
Dostupné z WWW: <www.stapro.cz>.

[24] SCOVECO [online]. 2013 [cit. 2013-04-02].
Dostupné z WWW: <www.scoveco.cz>.

[25] Skylink [online]. 2013 [cit. 2013-04-02].
Dostupné z WWW: <www.skylink.cz>.

[26] Raffo DM: Modeling software processes quantitatively and assessing the impact of potential process changes on process performance, 1996. Ph.D. thesis, Carnegie Mellon University.

[27] George S. Fishman. Discrete-Event Simulation: Modeling, Programming, and Analysis. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, Berlin, 2001.

[28] Marc I. Kellner, Raymond J. Madachy, and David M. Raffo. Software process simulation modeling: Why? What? How? The Journal of Systems and Software, 46(2–3):91–105, April 1999.

Zoznam príloh

Súčasťou DP je CD.

Adresárová štruktúra priloženého CD:

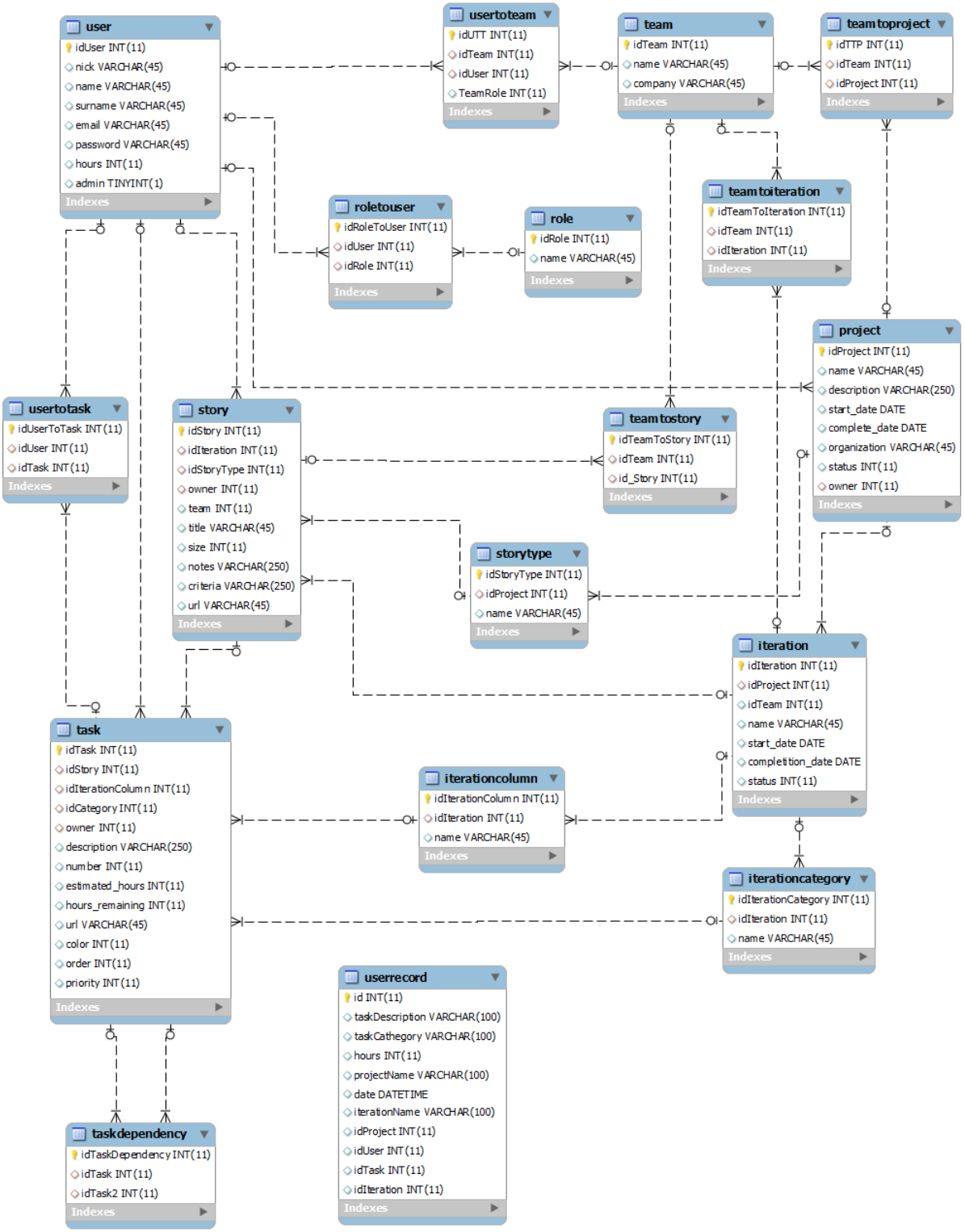
/text – text diplomovej práce v elektronickej podobe

/docs – ostatná dokumentácia k práci

/source – zdrojové kódy

/bin – skompilovaná aplikácia pre simuláciu

Príloha.A: Databázový model



Príloha.B: Business Use Cases

Business Use Cases: Riadenie vývoja softwaru s aplikačnou podporou	
BUC Skupina: Správa projektov	
Dôvod zoskupenia	Skupina zahŕňa služby pre prácu s projektom.
Aktér(i)	Administrátor
Poskytnuté služby	• Vytvorenie projektu • Zmena informácií o projekte
BUC Skupina: Správa užívateľov	
Dôvod zoskupenia	Skupina zahŕňa služby pre správu užívateľov.
Aktér(i)	Administrátor
Poskytnuté služby	• Vytvorenie účtu • Zrušenie účtu
BUC Skupina: Správa iterácií	
Dôvod zoskupenia	Skupina zahŕňa služby pre správu iterácií.
Aktér(i)	Administrátor
Poskytnuté služby	• Vytvorenie iterácie
BUC Skupina: Správa obsahu iterácie	
Dôvod zoskupenia	Skupina zahŕňa služby pre správu User Stories a taktiež služby pre správu úloh.
Aktér(i)	Administrátor, Užívateľ
Poskytnuté služby	• Vytvorenie User Story • Vytvorenie novej úlohy • Zmena informácií o úlohe

Príloha.C: Zoznam rolí

Procesné role	Product Owner, Scrum Master, Tím
Systémové role	Administrátor, Užívateľ

Product Owner	Vlastník projektu. V Scrume zastáva pozíciu zadávateľa projektu, za ktorého úspešné dokončenie je zodpovedný.
Scrum Master	Človek ktorý dohliada na správne dodržiavanie procesu v rámci pravidiel Scrumu.
Tím	Reprezentuje vývojárov pracujúcich na projekte. Tím je samo-riadiaci a každá člen tímu je rovnocenný.
Administrátor	Človek v roli administrátor má umožnenú kompletnú správu aplikácie a sú preňho prístupné všetky položky.
Užívateľ	Človek ktorého práva sú pri práci s aplikáciou obmedzené. Nemá prístup k evidencii rolí, tímov a užívateľov. Taktiež nevidí úlohy ktoré mu nie sú priradené.

Príloha.D: BUC1

BUC1: Vytvorenie projektu	
Primárny aktér	Administrátor
Biznis služba	
Udalosť	Vytvorenie nového projektu Administrátorom.
Od služby sa očakáva	Vytvorenie nového projektu, v prípade že má Administrátor k dispozícii zadanie projektu.
Biznis proces	
Vedľajší aktér(i)	Product Owner
Vstupná podmienka	Administrátor musí mať dispozícii zadanie projektu.
Spúšťač	Požiadavka na vytvorenie nového projektu.
Hlavný úspešný scenár	• Administrátor zadá aplikácií požiadavku pre vytvorenie nového projektu. • Aplikácia zobrazí formulár s možnosťou zadania informácií o projekte. • Administrátor vyplní informácie o projekte. • Administrátor zadá aplikácií požiadavku pre uloženie nového projektu. • Aplikácia uloží projekt so zadanými informáciami.
Očakávaný výsledok	Nový projekt je vytvorený.

Príloha.E: BUC2

BUC2: Zmena informácií o projekte	
Primárny Aktér	Administrátor
Biznis služba	
Udalosť	Administrátor dostane požiadavku pre zmenu informácií o projekte.
Od služby sa očakáva	Zmena informácií o existujúcom projekte.
Biznis proces	
Vedľajší aktér(i)	Product Owner
Vstupná podmienka	Administrátor musí mať dispozícii existujúci projekt.
Spúšťač	Požiadavka na zmenu informácií existujúceho projektu.
Hlavný úspešný scenár	• Administrátor zadá aplikácií požiadavku pre zmenu informácií o projekte. • Aplikácia zobrazí formulár s možnosťou zadania informácií o užívateľovi. • Administrátor zmení požadované informácie, prípadne ich doplní alebo odstráni. • Administrátor zadá aplikácií požiadavku pre uloženie projektu s aktuálne zadanými informáciami. • Aplikácia uloží projekt so zadanými informáciami.
Očakávaný výsledok	Informácie o projekte sú zmenené.

Príloha.F: BUC3

BUC3: Vytvorenie účtu	
Primárny Aktér	Administrátor
Biznis služba	
Udalosť	Administrátor dostane požiadavku pre vytvorenie nového účtu.
Od služby sa očakáva	Vytvorenie nového účtu.
Biznis proces	
Účastníci	Užívateľ
Vstupná podmienka	Administrátor musí mať dispozícii informácie o užívateľovi.
Spúšťač	Požiadavka na otvorenie nového účtu.
Hlavný úspešný scenár	• Administrátor zadá aplikácií požiadavku pre otvorenie nového účtu. • Aplikácia zobrazí formulár s možnosťou zadania informácií o užívateľovi. • Administrátor vyplní požadované informácie. • Administrátor zadá aplikácií požiadavku pre uloženie nového užívateľa so zadanými informáciami. • Aplikácia uloží užívateľa so zadanými informáciami.
Očakávaný výsledok	Nový účet je vytvorený.

Príloha.G: BUC4

BUC4: Zrušenie účtu	
Primárny Aktér	Administrátor
Biznis služba	
Udalosť	Administrátor dostane požiadavku pre zrušenie účtu.
Od služby sa očakáva	Zrušenie existujúceho účtu.
Biznis proces	
Vedľajší aktér	Užívateľ
Vstupná podmienka	Administrátor musí mať dispozícii existujúci účet.
Spúšťač	Požiadavka na zrušenie existujúceho účtu.
Hlavný úspešný scenár	• Administrátor zadá aplikácií požiadavku pre zobrazenie zoznamu existujúcich účtov. • Aplikácia zobrazí zoznam existujúcich účtov. • Administrátor zadá aplikácii požiadavku pre zrušenie vybratého účtu. • Aplikácia si vyžiada povolenie Administrátora pre zrušenie účtu. • Administrátor povolí zrušenie účtu. • Aplikácia odstráni zvolený účet.
Očakávaný výsledok	Zvolený účet bol odstránený.

Príloha.H: BUC5

BUC5: Vytvorenie iterácie	
Primárny Aktér	Administrátor
Biznis služba	
Udalosť	Vytvorenie iterácie v projekte.
Od služby sa očakáva	Vytvorenie novej iterácie.
Biznis proces	
Účastníci	Tím
Vstupné podmienky	Administrátor musí mať dispozícii informácie o iterácii. Administrátor musí mať k dispozícii existujúci projekt.
Spúšťač	Požiadavka na vytvorenie novej iterácie.
Hlavný úspešný scenár	• Administrátor zadá aplikácií požiadavku pre vytvorenie novej iterácie. • Aplikácia zobrazí formulár s možnosť zadania informácii o iterácii. • Administrátor zadá požadované informácie. • Administrátor zadá aplikácii požiadavku pre uloženie novej iterácie so zadanými informáciami. • Aplikácia uloží novú iteráciu.
Očakávaný výsledok	Nová iterácia je vytvorená.

Príloha.I: BUC6

BUC6: Vytvorenie User Story	
Primárny Aktér	Administrátor
Biznis služba	
Udalosť	Vytvorenie User Story v iterácii.
Od služby sa očakáva	Vytvorenie nového User Story.
Biznis proces	
Účastníci	Tím
Vstupné podmienky	Administrátor musí mať dispozícii informácie o User Story Administrátor musí mať k dispozícii existujúcu iteráciu.
Spúšťač	Požiadavka na vytvorenie nového User Story.
Hlavný úspešný scenár	• Administrátor zadá aplikácií požiadavku pre vytvorenie nového User Story. • Aplikácia zobrazí formulár s možnosť zadania informácii o User Story. • Administrátor zadá požadované informácie. • Administrátor zadá aplikácii požiadavku pre uloženie nového User Story so zadanými informáciami. • Aplikácia uloží nový User Story.
Očakávaný výsledok	Nový User Story je vytvorený.

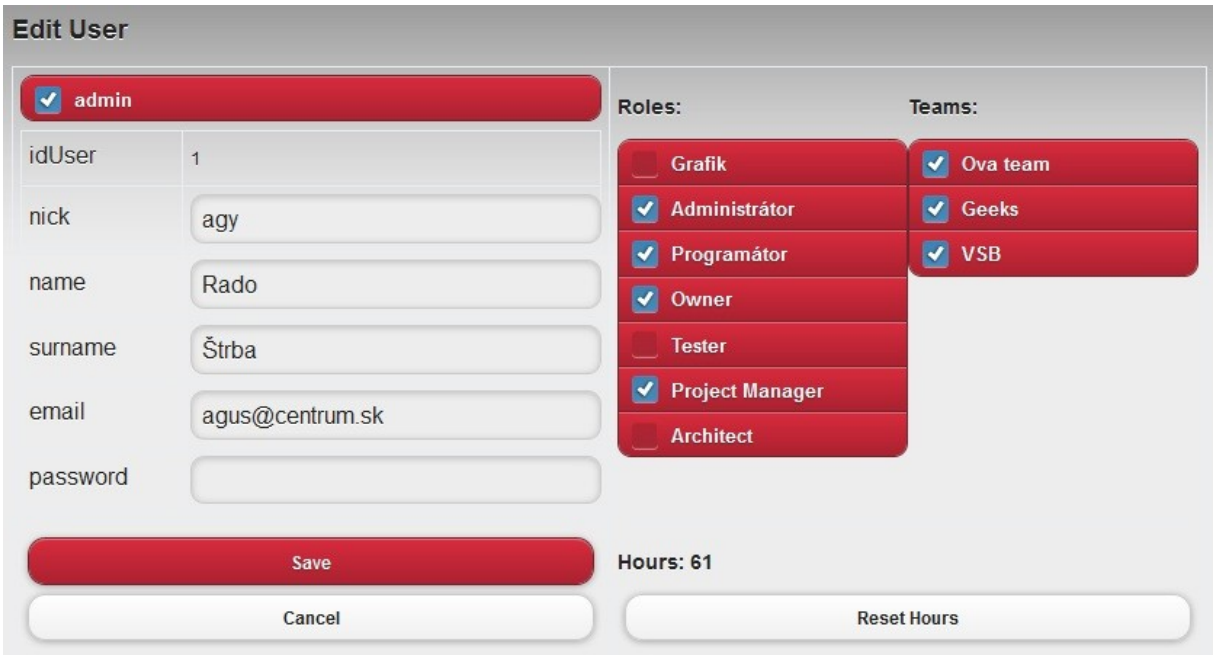
Príloha.J: BUC7

BUC7: Vytvorenie úlohy	
Primárny Aktér	Administrátor
Biznis služba	
Udalosť	Užívateľ chce zmeniť informácie o úlohe.
Od služby sa očakáva	Vytvorenie novej úlohy.
Biznis proces	
Účastníci	Tím
Vstupné podmienky	Administrátor musí mať dispozícii informácie o úlohe. Administrátor musí mať k dispozícii existujúci User Story.
Spúšťač	Požiadavka na vytvorenie novej úlohy.
Hlavný úspešný scenár	• Administrátor zadá aplikácií požiadavku pre vytvorenie novej úlohy. • Aplikácia zobrazí formulár s možnosť zadania informácii o úlohe. • Administrátor zadá požadované informácie. • Administrátor zadá aplikácii požiadavku pre uloženie novej úlohy so zadanými informáciami. • Aplikácia uloží novú úlohu.
Očakávaný výsledok	Nová úloha je vytvorená.

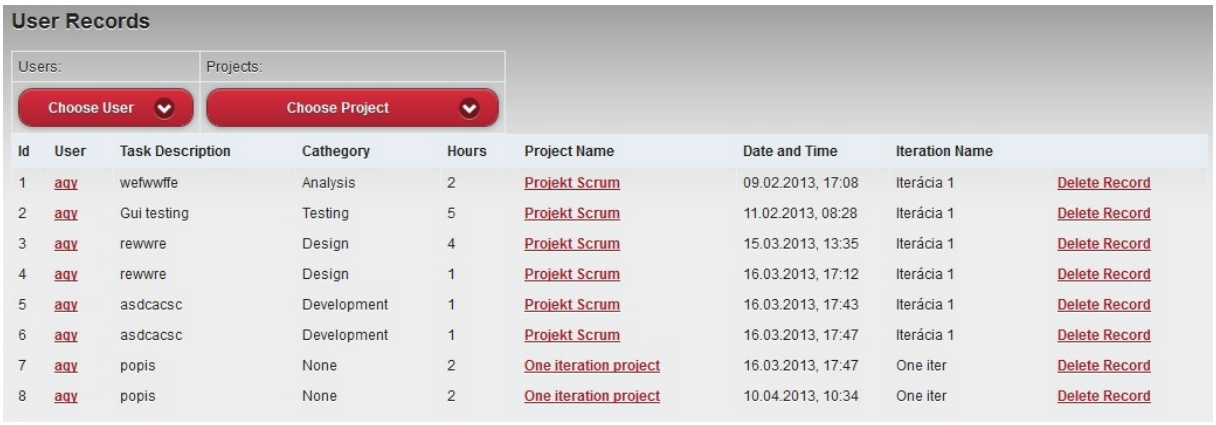
Príloha.K: BUC8

BUC8: Zmena informácií o úlohe	
Primárny Aktér	Užívateľ
Biznis služba	
Udalosť	
Od služby sa očakáva	Zmena informácií o existujúcej úlohe.
Biznis proces	
Vedľajší aktér(i)	Tím
Vstupná podmienka	Užívate musí mať dispozícii existujúcu úlohu.
Spúšťač	Požiadavka na zmenu informácií existujúcej úlohe.
Hlavný úspešný scenár	• Užívateľ zadá aplikácií požiadavku pre zmenu informácií o úlohe. • Aplikácia zobrazí formulár s možnosťou zadania informácií o úlohe. • Administrátor zmení požadované informácie, prípadne ich doplní alebo odstráni. • Administrátor zadá aplikácií požiadavku pre uloženie úlohy s aktuálne zadanými informáciami. • Aplikácia uloží úlohu so zadanými informáciami.
Očakávaný výsledok	Informácie o úlohe sú zmenené.

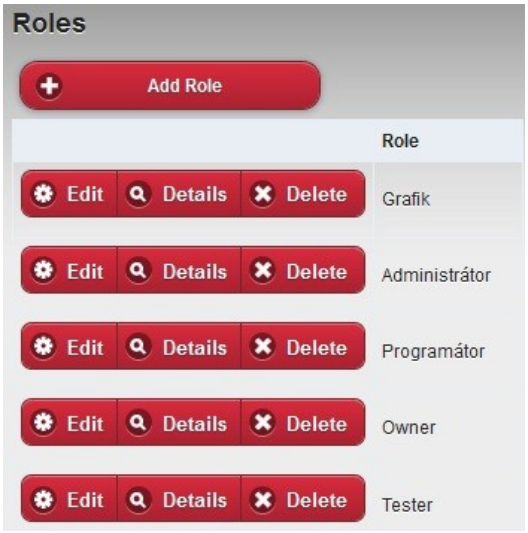
Príloha.L: Prehľad a úprava užívateľov



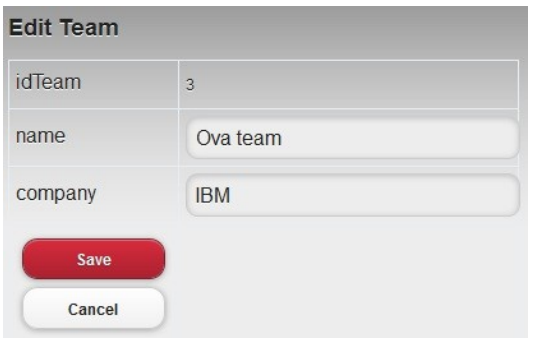
Príloha.M: Užívateľské záznamy



Príloha.N: Prehľad a úprava rolí



Príloha.O: Prehľad a úprava tímov



Príloha.P: Úprava vytvorenej úlohy

Príloha.Q: Potvrdenie o zaškolení kľúčového zástupcu spoločnosti STAPRO, s.r.o.

Potvrzení o provedení zaškolení na téma:

Používání aplikace softwarové podpory SCRUM,
pro společnost STAPRO, s.r.o.

Školitel:
Bc. Radoslav Štrba

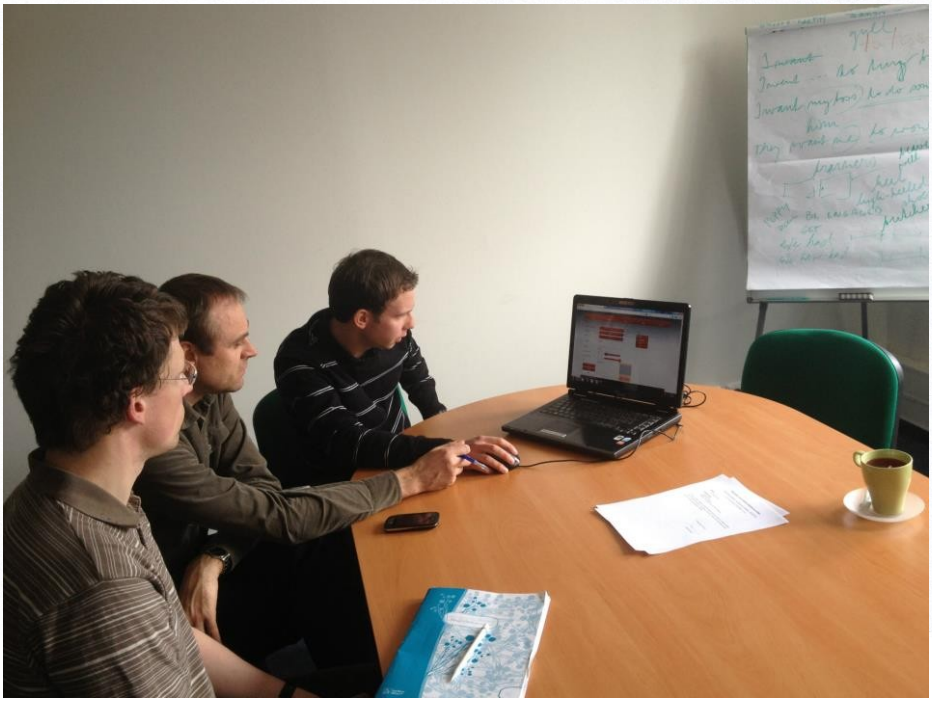
Zaškolená osoba:
Ing. Jan Bičiště,
Ředitel úseku vývoje SW,
STAPRO s.r.o., Pernštýnské nám. 51, 530 02 Pardubice

Potvrzuji, že došlo k zaškolení klíčového zástupce společnosti Stapro pro oblast metodiky vývoje SW SCRUM společně s metodikami o správném dodržování softwarového procesu, který byl navržený pro společnost STAPRO.

V Ostravě dne 10.4.2013:

.....

(zaškolená osoba)



Príloha.R: Potvrdenie o zaškolení kľúčového zástupcu spoločnosti SCOVECO, s.r.o.

Potvrzení o provedení zaškolení na téma:

Používaní aplikace softwarové podpory SCRUM,
pro společnost SCOVECO, s.r.o.

Školitel:
Bc. Radoslav Štrba

Zaškolená osoba:
Ing. Zdeněk Velart, Ph.D. ,
Ředitel vývoje,
SCOVECO, s.r.o., Františka Lýska 1605/3, 700 30, Ostrava - Bělský Les

Potvrzuji, že došlo k zaškolení klíčového zástupce společnosti SCOVECO pro oblast metodiky vývoje SW SCRUM společně s metodikami o správném dodržovaní softwarového procesu, který byl navržený pro společnost SCOVECO.

V Ostravě dne 15.4.2013:

.....

(zaškolená osoba)